ESRF Double Crystal Monochromator - Lookup Tables

Dehaeze Thomas

January 11, 2022

Contents

A Fast Jack is composed of one stepper motor and a piezoelectric stack in series.

The stepper motor is directly driving (i.e. without a reducer) a ball screw with a pitch of 1mm (i.e. 1 stepper motor turn makes a 1mm linear motion).

The stepper motor is doing the coarse displacement while the piezoelectric stack is only there to compensate all the motion errors of the stepper motor.

A Lookup Tables (LUT) is used to compensate for **repeatable** errors of the stepper motors. This has several goals:

- Reduce errors down to the stroke of the piezoelectric stack actuator
- Reduce errors above the bandwidth of the feedback controller

1 Schematic

In order to measure the errors induced by the fast jacks, we have to make some scans, and measure simultaneously:

- The wanted fast jack position: signal/step sent by the IcePAP
- The actual (measured) position

The experimental setup to perform this is shown in Figure 1.1.

The procedure is the following:

- A scan on the Bragg angle θ is generated from Bliss
- Reference paths $[r_{u_r},\ r_{u_h},\ r_d]$ are sent to the IcePAP
- Initially, the LUT inside the IcePAP is not changing the reference path
- The IcePAP generates some steps $[u_{u_r},\ u_{u_h},\ u_d]$ that are sent to the fast jacks
- The motion of the crystals $[d_z,\ r_y,\ r_x]$ is measured with the interferometers and computed in the Speedgoat
- Finally, the corresponding motion $[d_{u_r},\ r_{u_h},\ r_d]$ of the fast jack is computed afterwards in BLISS

The measured motion of the fast jacks $[d_{u_r},\ r_{u_h},\ r_d]$ can be compared with the IcePAP steps $[u_{u_r},\ u_{u_h},\ u_d]$ in order to create the LUT inside the IcePAP.

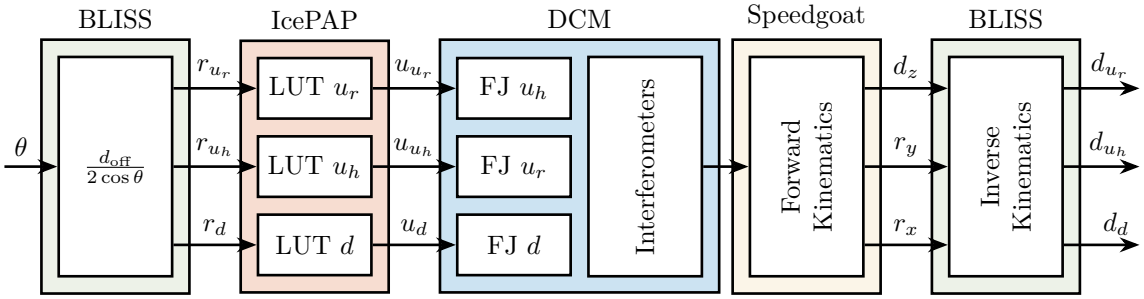

Figure 1.1: Block diagram of the experiment to create the Lookup Table

2 First Analysis

2.1 Patterns in the Fast Jack motion errors

In order to understand what should be the "sampling distance" for the lookup table of the stepper motor, we have to analyze the displacement errors induced by the stepper motor.

Let's load the measurements of one bragg angle scan without any LUT.

Matlab

```
%% Load Data of the new LUT method
ol_bragg = (pi/180)*1e-5*double(h5read('Qutools_test_0001.h5','/33.1/instrument/trajmot/data')); % Bragg angle [rad]
ol_dzw   = 10.5e-3./(2*cos(ol_bragg)); % Wanted distance between crystals [m]

ol_dz    = 1e-9*double(h5read('Qutools_test_0001.h5','/33.1/instrument/xtal_111_dz_filter/data'));  % Dz distance between
↪  crystals [m]
ol_dry   = 1e-9*double(h5read('Qutools_test_0001.h5','/33.1/instrument/xtal_111_dry_filter/data')); % Ry [rad]
ol_drx   = 1e-9*double(h5read('Qutools_test_0001.h5','/33.1/instrument/xtal_111_drx_filter/data')); % Rx [rad]

ol_t = 1e-6*double(h5read('Qutools_test_0001.h5','/33.1/instrument/time/data')); % Time [s]

ol_ddz   = ol_dzw-ol_dz; % Distance Error between crystals [m]
```

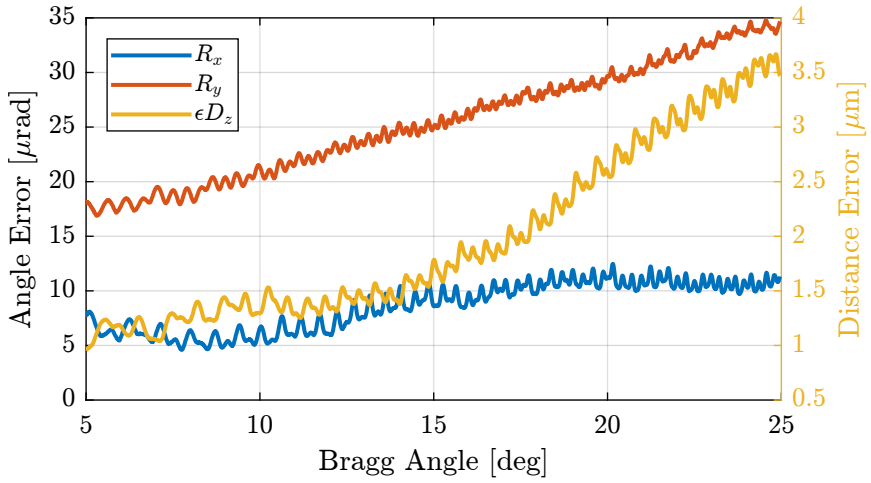

Figure 2.1: Orientation and Distance error of the Crystal measured by the interferometers

Now let's convert the errors from the frame of the crystal to the frame of the fast jacks (inverse kinematics problem) using the Jacobian matrix.

Matlab

```
%% Compute Fast Jack position errors
% Jacobian matrix for Fast Jacks and 111 crystal
J_a_111 = [1,  0.14, -0.1525
           1,  0.14,  0.0675
```

```
          1, -0.14,  0.0425];

ol_de_111 = [ol_ddz'; ol_dry'; ol_drx'];

% Fast Jack position errors
ol_de_fj = J_a_111*ol_de_111;

ol_fj_ur = ol_de_fj(1,:);
ol_fj_uh = ol_de_fj(2,:);
ol_fj_d  = ol_de_fj(3,:);
```

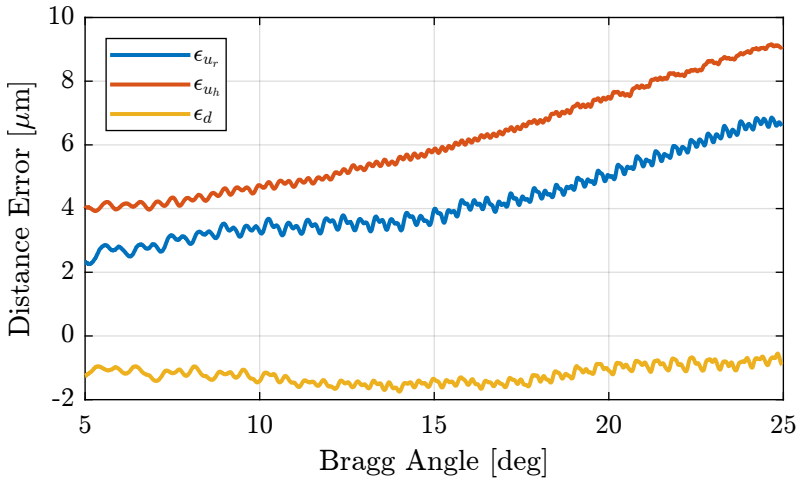

Figure 2.2: Estimated motion errors of the fast jacks during the scan

Let's now identify this pattern as a function of the fast-jack position.

As we want to done frequency Fourier transform, we need to have uniform sampling along the fast jack position. To do so, the function `resample` is used.

Matlab
```
Xs = 0.1e-6; % Sampling Distance [m]

%% Re-sampled data with uniform spacing [m]
ol_fj_ur_u = resample(ol_fj_ur, ol_dzw, 1/Xs);
ol_fj_uh_u = resample(ol_fj_uh, ol_dzw, 1/Xs);
ol_fj_d_u  = resample(ol_fj_d,  ol_dzw, 1/Xs);

ol_fj_u = Xs*[1:length(ol_fj_ur_u)]; % Sampled Jack Position
```

The result is shown in Figure 2.3.

Let's now perform a Power Spectral Analysis of the measured displacement errors of the Fast Jack.

Matlab
```
% Hanning Windows with 250um width
win = hanning(floor(400e-6/Xs));

% Power Spectral Density [m2/(1/m)]
[S_fj_ur, f] = pwelch(ol_fj_ur_u-mean(ol_fj_ur_u), win, 0, [], 1/Xs);
[S_fj_uh, ~] = pwelch(ol_fj_uh_u-mean(ol_fj_uh_u), win, 0, [], 1/Xs);
[S_fj_d,  ~] = pwelch(ol_fj_d_u -mean(ol_fj_d_u ),  win, 0, [], 1/Xs);
```

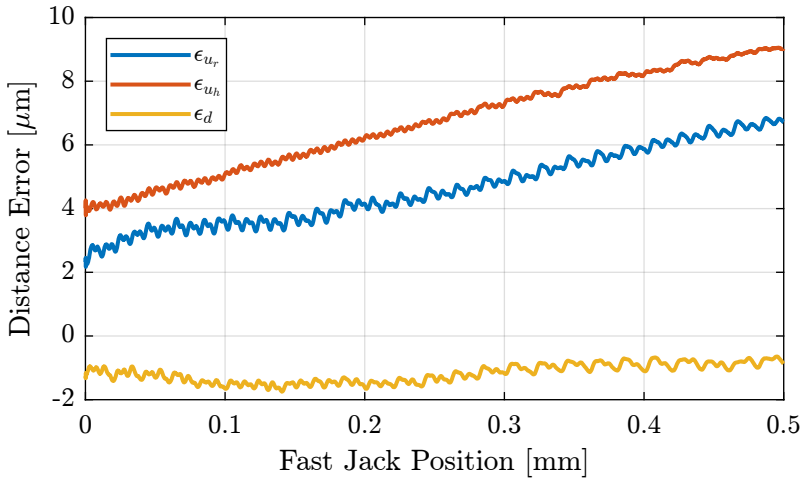

Figure 2.3: Position error of fast jacks as a function of the fast jack motion

As shown in Figure 2.4, we can see a fundamental "reciprocal length" of $5 \cdot 10^4\,[1/m]$ and its harmonics. This corresponds to a length of $\frac{1}{5 \cdot 10^4} = 20\,[\mu m]$.

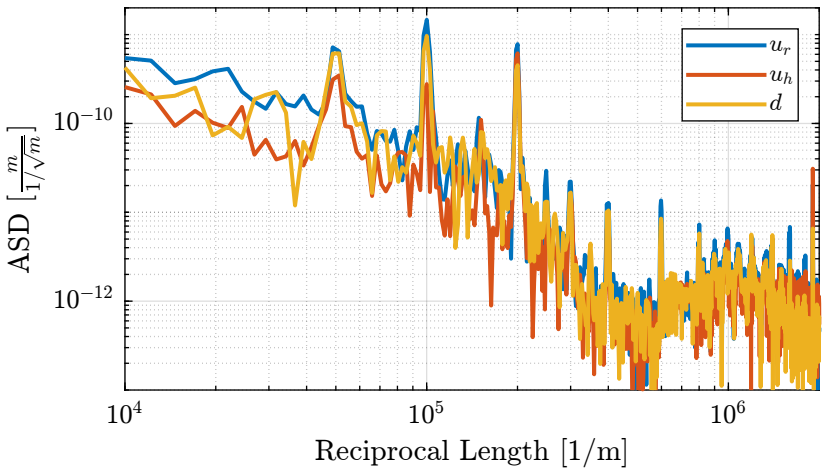

Figure 2.4: Spectral content of the error as a function of the reciprocal length

Instead of looking at that as a function of the reciprocal length, we can look at it as a function of the spectral distance (Figure 2.5).

We see that the errors have a pattern with "spectral distances" equal to $5\,[\mu m]$, $10\,[\mu m]$, $20\,[\mu m]$ and smaller harmonics.

Let's try to understand these results. One turn of the stepper motor corresponds to a vertical motion of 1mm. The stepper motor has 50 pairs of poles, therefore one pair of pole corresponds to a motion of $20\,[\mu m]$ which is the fundamental "spectral distance" we observe.

Matlab
```
CPS_ur = flip(-cumtrapz(flip(f), flip(S_fj_ur)));
CPS_uh = flip(-cumtrapz(flip(f), flip(S_fj_uh)));
CPS_d  = flip(-cumtrapz(flip(f), flip(S_fj_d)));
```

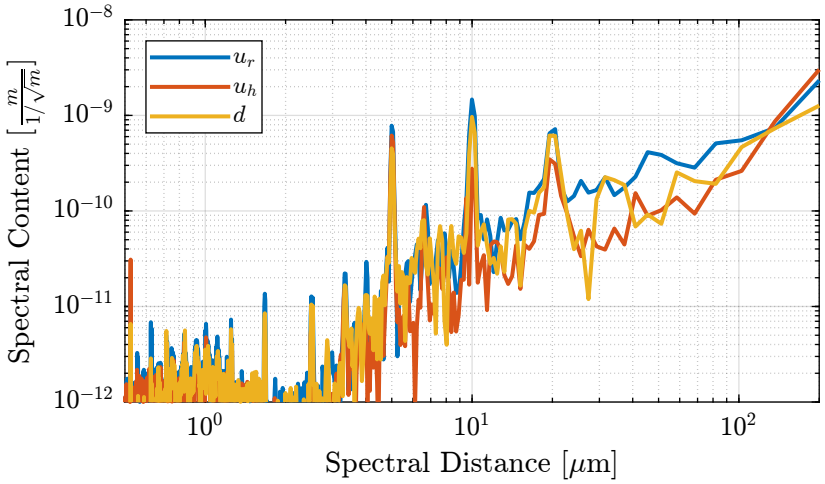

Figure 2.5: Spectral content of the error as a function of the spectral distance

From Figure 2.6, we can see that if the motion errors with a period of $5\,[\mu m]$ and $10\,[\mu m]$ can be dealt with the lookup table, this will reduce a lot the positioning errors of the fast jack.

Matlab

```
%% Cumulative Spectrum
figure;
hold on;
plot(1e6./f, sqrt(CPS_ur), 'DisplayName', '$u_r$');
plot(1e6./f, sqrt(CPS_uh), 'DisplayName', '$u_j$');
plot(1e6./f, sqrt(CPS_d),  'DisplayName', '$d$');
hold off;
set(gca, 'xscale', 'log'); set(gca, 'yscale', 'log');
xlabel('Spectral Distance [$\mu m$]'); ylabel('Cumulative Spectrum [$m$]')
xlim([1, 500]); ylim([1e-9, 1e-5]);
legend('location', 'northwest');
```

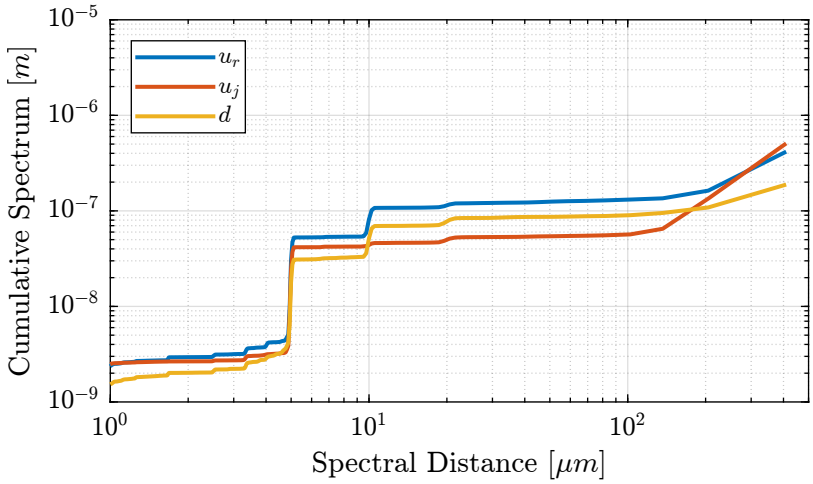

Figure 2.6: Cumulative spectrum from small spectral distances to large spectral distances

2.2 Experimental Data - Current Method

The current used method is an iterative one.

Matlab

```
%% Load Experimental Data
ol_bragg    = double(h5read('first_beam_0001.h5','/31.1/instrument/trajmot/data'));
ol_drx      = h5read('first_beam_0001.h5','/31.1/instrument/xtal_111_drx_filter/data');

lut_1_bragg = double(h5read('first_beam_0001.h5','/32.1/instrument/trajmot/data'));
lut_1_drx   = h5read('first_beam_0001.h5','/32.1/instrument/xtal_111_drx_filter/data');

lut_2_bragg = double(h5read('first_beam_0001.h5','/33.1/instrument/trajmot/data'));
lut_2_drx   = h5read('first_beam_0001.h5','/33.1/instrument/xtal_111_drx_filter/data');

lut_3_bragg = double(h5read('first_beam_0001.h5','/34.1/instrument/trajmot/data'));
lut_3_drx   = h5read('first_beam_0001.h5','/34.1/instrument/xtal_111_drx_filter/data');

lut_4_bragg = double(h5read('first_beam_0001.h5','/36.1/instrument/trajmot/data'));
lut_4_drx   = h5read('first_beam_0001.h5','/36.1/instrument/xtal_111_drx_filter/data');
```

The relative orientation of the two `111` mirrors in the x directions are compared in Figure 2.7 for several iterations. We can see that after the first iteration, the orientation error has an opposite sign as for the case without LUT.

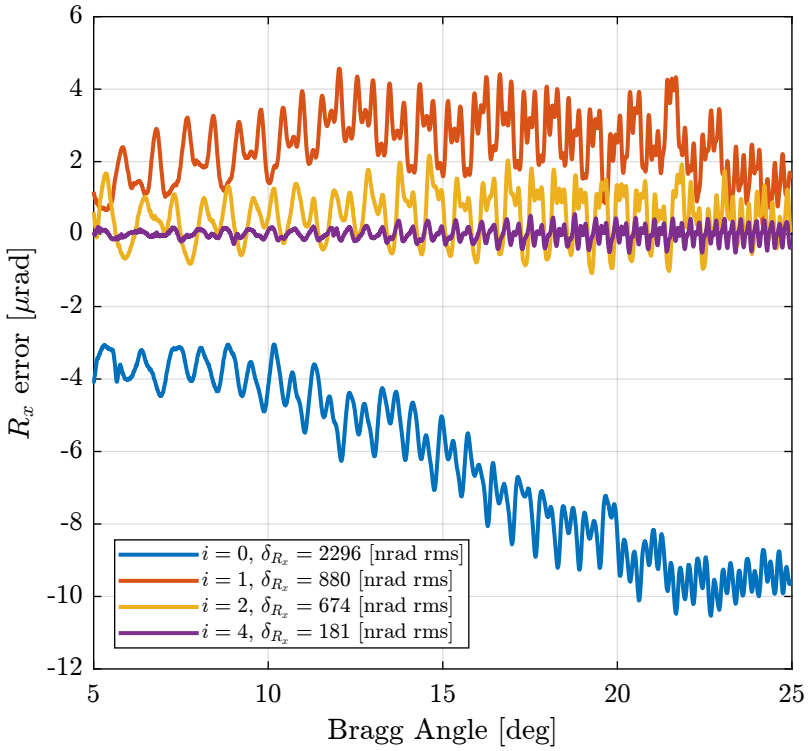

Figure 2.7: R_x error with the current LUT method

2.3 Simulation

In this section, we suppose that we are in the frame of one fast jack (all transformations are already done), and we wish to create a LUT for one fast jack.

Let's say with make a Bragg angle scan between 10deg and 60deg during 100s.

```Matlab
Fs = 10e3; % Sample Frequency [Hz]
t = 0:1/Fs:10; % Time vector [s]
theta = linspace(10, 40, length(t)); % Bragg Angle [deg]
```

The IcePAP steps are following the theoretical formula:

$$d_z = \frac{d_{\text{off}}}{2\cos\theta} \tag{2.1}$$

with θ the bragg angle and $d_{\text{off}} = 10\,mm$.

The motion to follow is then:

```Matlab
perfect_motion = 10e-3./(2*cos(theta*pi/180)); % Perfect motion [m]
```

And the IcePAP is generated those steps:

```Matlab
icepap_steps = perfect_motion; % IcePAP steps measured by Speedgoat [m]
```

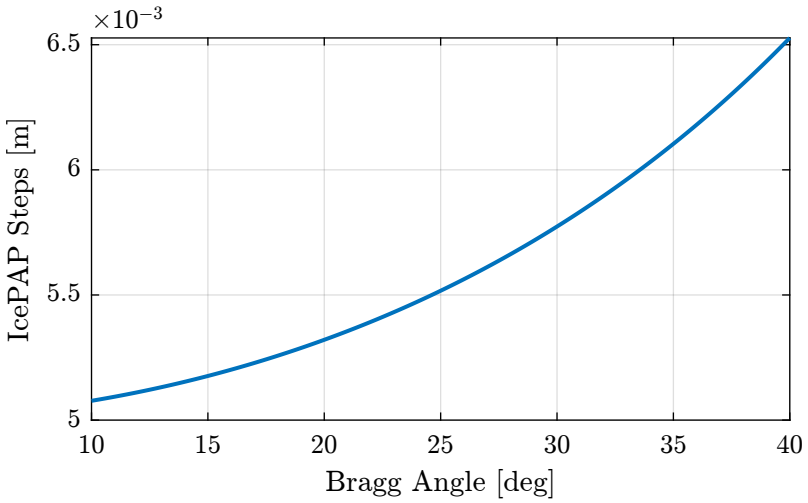

Figure 2.8: IcePAP Steps as a function of the Bragg Angle

Then, we are measuring the motion of the Fast Jack using the Interferometer. The motion error is larger than in reality to be angle to see it more easily.

```matlab
motion_error = 100e-6*sin(2*pi*perfect_motion/1e-3); % Error motion [m]

measured_motion = perfect_motion + motion_error; % Measured motion of the Fast Jack [m]
```

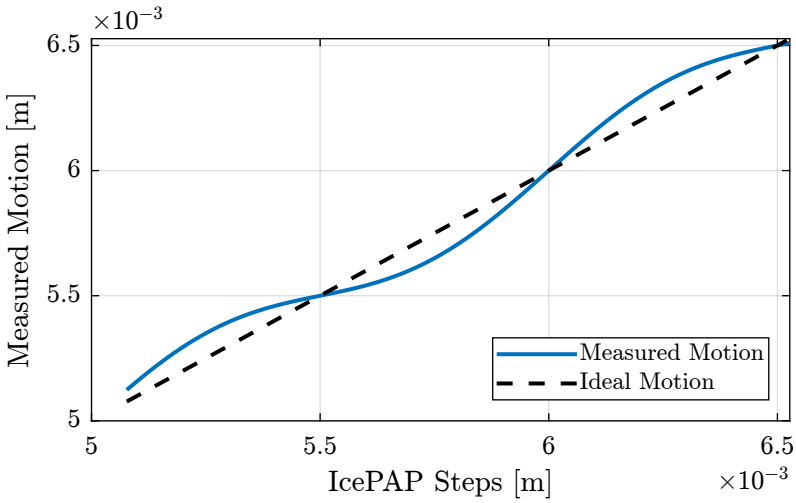

Figure 2.9: Measured motion as a function of the IcePAP Steps

Let's now compute the lookup table. For each micrometer of the IcePAP step, another step is associated that correspond to a position closer to the wanted position.

```matlab
%% Get range for the LUT
% We correct only in the range of tested/measured motion
lut_range = round(1e6*min(icepap_steps)):round(1e6*max(icepap_steps)); % IcePAP steps [um]

%% Initialize the LUT
lut = zeros(size(lut_range));

%% For each um in this range
for i = 1:length(lut_range)
    % Get points indices where the measured motion is closed to the wanted one
    close_points = measured_motion > 1e-6*lut_range(i) - 500e-9 & measured_motion < 1e-6*lut_range(i) + 500e-9;
    % Get the corresponding closest IcePAP step
    lut(i) = round(1e6*mean(icepap_steps(close_points))); % [um]
end
```

The current LUT implementation is the following:

```matlab
motion_error_lut = zeros(size(lut_range));
for i = 1:length(lut_range)
    % Get points indices where the icepap step is close to the wanted one
    close_points = icepap_steps > 1e-6*lut_range(i) - 500e-9 & icepap_steps < 1e-6*lut_range(i) + 500e-9;
    % Get the corresponding motion error
    motion_error_lut(i) = lut_range(i) + (lut_range(i) - round(1e6*mean(measured_motion(close_points)))); % [um]
end
```

Let's compare the two Lookup Table in Figure 2.11.

If we plot the "corrected steps" for all steps for both methods, we clearly see the difference (Figure 2.12).

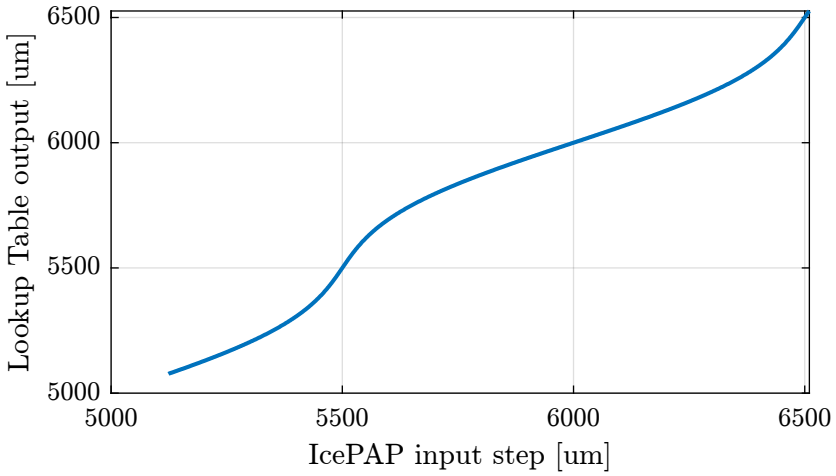

Figure 2.10: Generated Lookup Table

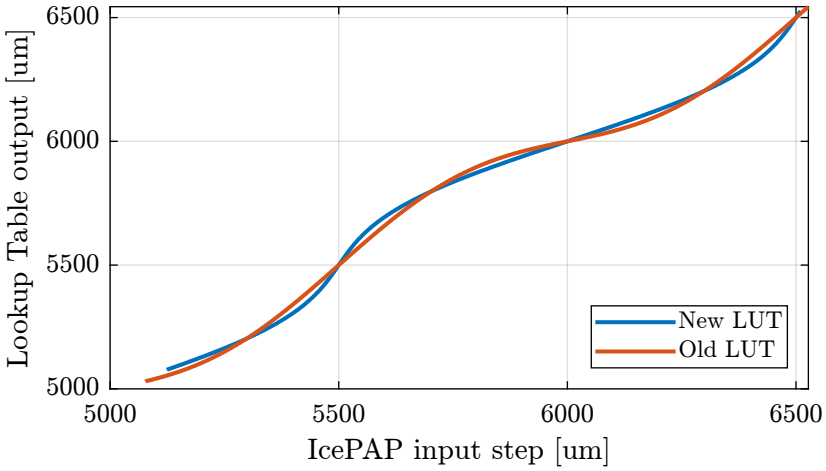

Figure 2.11: Comparison of the two lookup tables

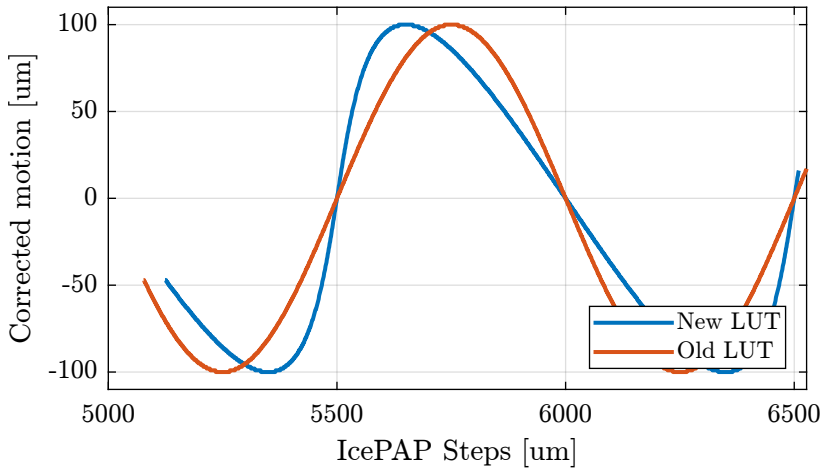

Figure 2.12: LUT correction and motion error as a function of the IcePAP steps

Let's now implement both LUT to see which implementation is correct.

```Matlab
motion_new = zeros(size(icepap_steps_output_new));
motion_old = zeros(size(icepap_steps_output_old));

for i = 1:length(icepap_steps_output_new)
    [~, i_step] = min(abs(icepap_steps_output_new(i) - 1e6*icepap_steps));
    motion_new(i) = measured_motion(i_step);

    [~, i_step] = min(abs(icepap_steps_output_old(i) - 1e6*icepap_steps));
    motion_old(i) = measured_motion(i_step);
end
```

The output motion with both LUT are shown in Figure 2.13. It is confirmed that the new LUT is the correct one. Also, it is interesting to note that the old LUT gives an output motion that is above the ideal one, as was seen during the experiments.

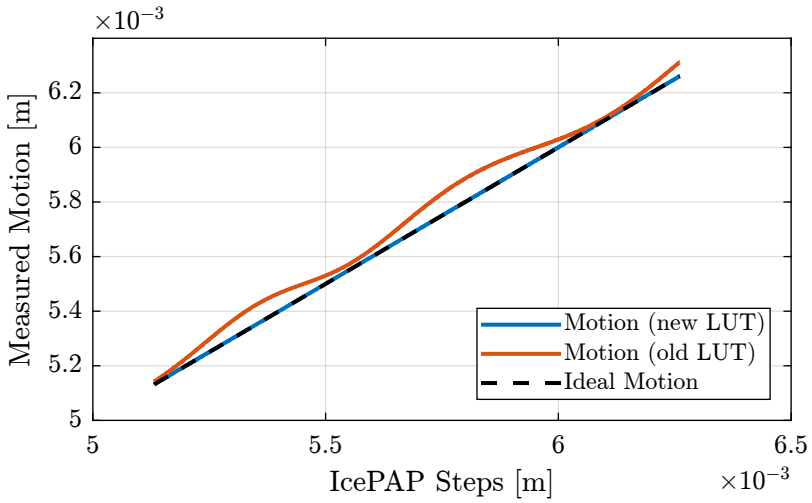

Figure 2.13: Comparison of the obtained motion with new and old LUT

2.4 Experimental Data - Proposed method (BLISS first implementation)

The new proposed method has been implemented and tested.

The result is shown in Figure 2.14. After only one iteration, the result is close to the previous method.

```Matlab
%% Load Data of the new LUT method
ol_new_bragg = double(h5read('Qutools_test_0001.h5','/33.1/instrument/trajmot/data'));
ol_new_drx   = h5read('Qutools_test_0001.h5','/33.1/instrument/xtal_111_drx_filter/data');

lut_new_bragg = double(h5read('Qutools_test_0001.h5','/34.1/instrument/trajmot/data'));
lut_new_drx   = h5read('Qutools_test_0001.h5','/34.1/instrument/xtal_111_drx_filter/data');
```

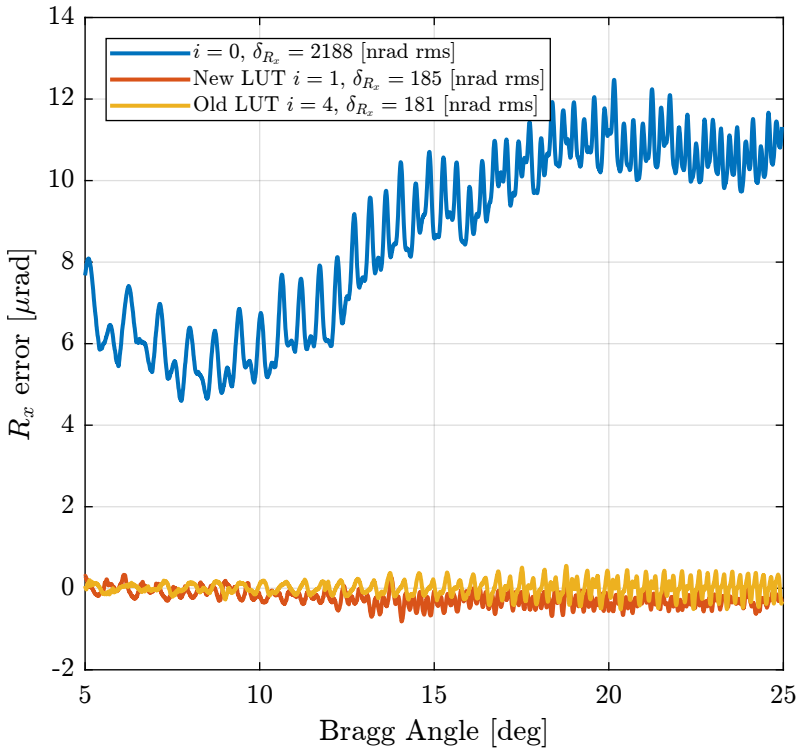

Figure 2.14: Comparison of the R_x error for the current LUT method and the proposed one

If we zoom on the 20deg to 25deg bragg angles, we can see that the new method has much less "periodic errors" as compared to the previous one which shows some patterns.

2.5 Comparison of the errors in the reciprocal length space

Matlab

```
%% Load Data of the new LUT method
ol_bragg = (pi/180)*1e-5*double(h5read('Qutools_test_0001.h5','/33.1/instrument/trajmot/data'));
ol_dz    = 1e-9*double(h5read('Qutools_test_0001.h5','/33.1/instrument/xtal_111_dz_filter/data'));
ol_dry   = 1e-9*double(h5read('Qutools_test_0001.h5','/33.1/instrument/xtal_111_dry_filter/data'));
ol_drx   = 1e-9*double(h5read('Qutools_test_0001.h5','/33.1/instrument/xtal_111_drx_filter/data'));
ol_dzw   = 10.5e-3./(2*cos(ol_bragg)); % Wanted distance between crystals [m]
ol_t     = 1e-6*double(h5read('Qutools_test_0001.h5','/33.1/instrument/time/data')); % Time [s]
ol_ddz   = ol_dzw-ol_dz; % Distance Error between crystals [m]

lut_bragg = (pi/180)*1e-5*double(h5read('Qutools_test_0001.h5','/34.1/instrument/trajmot/data'));
lut_dz    = 1e-9*double(h5read('Qutools_test_0001.h5','/34.1/instrument/xtal_111_dz_filter/data'));
lut_dry   = 1e-9*double(h5read('Qutools_test_0001.h5','/34.1/instrument/xtal_111_dry_filter/data'));
lut_drx   = 1e-9*double(h5read('Qutools_test_0001.h5','/34.1/instrument/xtal_111_drx_filter/data'));
lut_dzw   = 10.5e-3./(2*cos(lut_bragg)); % Wanted distance between crystals [m]
lut_t     = 1e-6*double(h5read('Qutools_test_0001.h5','/34.1/instrument/time/data')); % Time [s]
lut_ddz   = lut_dzw-lut_dz; % Distance Error between crystals [m]
```

Matlab

```
%% Compute Fast Jack position errors
% Jacobian matrix for Fast Jacks and 111 crystal
J_a_111 = [1,  0.14, -0.1525
           1,  0.14,  0.0675
```

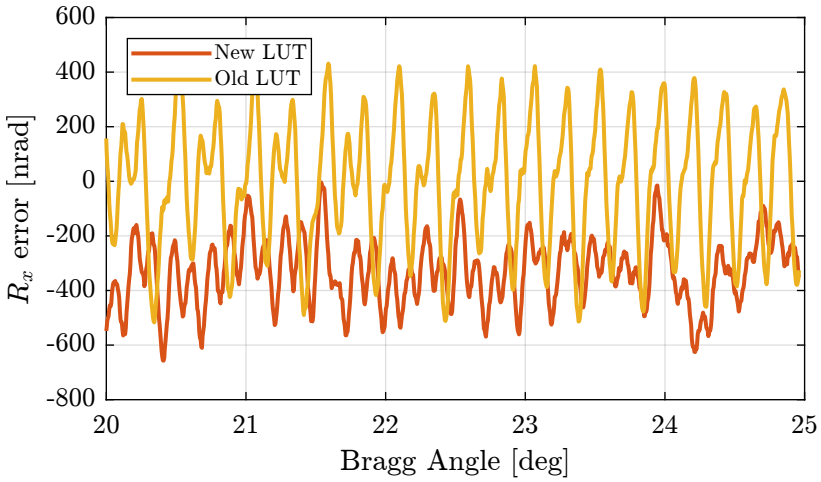

Figure 2.15: Comparison of the residual motion after old LUT and new LUT

```
           1, -0.14,  0.0425];

ol_de_111 = [ol_ddz'; ol_dry'; ol_drx'];

% Fast Jack position errors
ol_de_fj = J_a_111*ol_de_111;

ol_fj_ur = ol_de_fj(1,:);
ol_fj_uh = ol_de_fj(2,:);
ol_fj_d  = ol_de_fj(3,:);

lut_de_111 = [lut_ddz'; lut_dry'; lut_drx'];

% Fast Jack position errors
lut_de_fj = J_a_111*lut_de_111;

lut_fj_ur = lut_de_fj(1,:);
lut_fj_uh = lut_de_fj(2,:);
lut_fj_d  = lut_de_fj(3,:);
```

Matlab

```
Xs = 0.1e-6; % Sampling Distance [m]

%% Re-sampled data with uniform spacing [m]
ol_fj_ur_u = resample(ol_fj_ur, ol_dzw, 1/Xs);
ol_fj_uh_u = resample(ol_fj_uh, ol_dzw, 1/Xs);
ol_fj_d_u  = resample(ol_fj_d,  ol_dzw, 1/Xs);

ol_fj_u = Xs*[1:length(ol_fj_ur_u)]; % Sampled Jack Position

% Only take first 500um
ol_fj_ur_u = ol_fj_ur_u(ol_fj_u<0.5e-3);
ol_fj_uh_u = ol_fj_uh_u(ol_fj_u<0.5e-3);
ol_fj_d_u  = ol_fj_d_u (ol_fj_u<0.5e-3);
ol_fj_u    = ol_fj_u   (ol_fj_u<0.5e-3);
```

Matlab

```
%% Re-sampled data with uniform spacing [m]
lut_fj_ur_u = resample(lut_fj_ur, lut_dzw, 1/Xs);
lut_fj_uh_u = resample(lut_fj_uh, lut_dzw, 1/Xs);
lut_fj_d_u  = resample(lut_fj_d,  lut_dzw, 1/Xs);

lut_fj_u = Xs*[1:length(lut_fj_ur_u)]; % Sampled Jack Position
```

```
% Only take first 500um
lut_fj_ur_u = lut_fj_ur_u(lut_fj_u<0.5e-3);
lut_fj_uh_u = lut_fj_uh_u(lut_fj_u<0.5e-3);
lut_fj_d_u  = lut_fj_d_u (lut_fj_u<0.5e-3);
lut_fj_u    = lut_fj_u   (lut_fj_u<0.5e-3);
```

Matlab

```
% Hanning Windows with 250um width
win = hanning(floor(400e-6/Xs));

% Power Spectral Density [m2/(1/m)]
[S_ol_ur, f] = pwelch(ol_fj_ur_u-mean(ol_fj_ur_u), win, 0, [], 1/Xs);
[S_ol_uh, ~] = pwelch(ol_fj_uh_u-mean(ol_fj_uh_u), win, 0, [], 1/Xs);
[S_ol_d,  ~] = pwelch(ol_fj_d_u -mean(ol_fj_d_u ), win, 0, [], 1/Xs);

[S_lut_ur, ~] = pwelch(lut_fj_ur_u-mean(lut_fj_ur_u), win, 0, [], 1/Xs);
[S_lut_uh, ~] = pwelch(lut_fj_uh_u-mean(lut_fj_uh_u), win, 0, [], 1/Xs);
[S_lut_d,  ~] = pwelch(lut_fj_d_u -mean(lut_fj_d_u ), win, 0, [], 1/Xs);
```

As seen in Figure 2.16, the LUT as an effect only on spatial errors with a period of at least few μm. This is very logical considering the $1\,\mu m$ sampling of the LUT in the IcePAP.

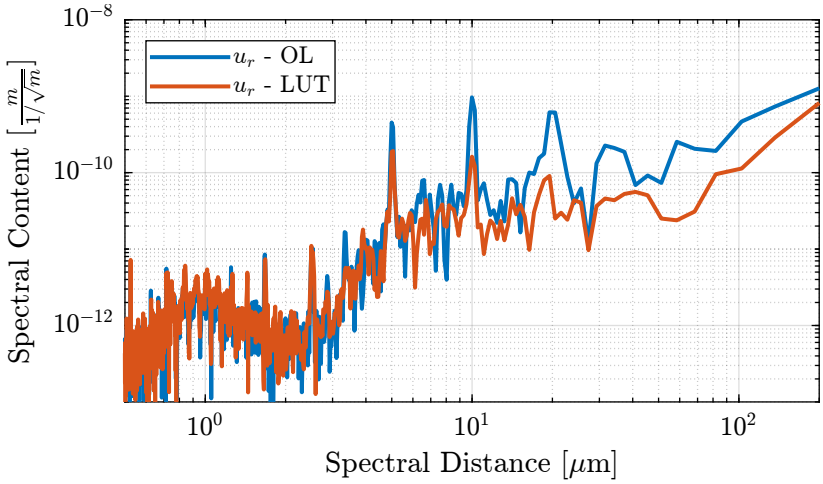

Figure 2.16: Effect of the LUT on the spectral content of the positioning errors

Let's now look at it in a cumulative way.

Matlab

```
CPS_ol_ur = flip(-cumtrapz(flip(f), flip(S_ol_ur)));
CPS_ol_uh = flip(-cumtrapz(flip(f), flip(S_ol_uh)));
CPS_ol_d  = flip(-cumtrapz(flip(f), flip(S_ol_d)));

CPS_lut_ur = flip(-cumtrapz(flip(f), flip(S_lut_ur)));
CPS_lut_uh = flip(-cumtrapz(flip(f), flip(S_lut_uh)));
CPS_lut_d  = flip(-cumtrapz(flip(f), flip(S_lut_d)));
```

Matlab

```
%% Cumulative Spectrum
figure;
hold on;
plot(1e6./f, sqrt(CPS_ol_ur) , 'DisplayName', '$u_r$ - OL');
plot(1e6./f, sqrt(CPS_lut_ur), 'DisplayName', '$u_r$ - LUT');
hold off;
```

```
set(gca, 'xscale', 'log'); set(gca, 'yscale', 'log');
xlabel('Spectral Distance [$\mu m$]'); ylabel('Cumulative Spectrum [$m$]')
xlim([1, 500]); ylim([1e-9, 1e-5]);
legend('location', 'northwest');
```

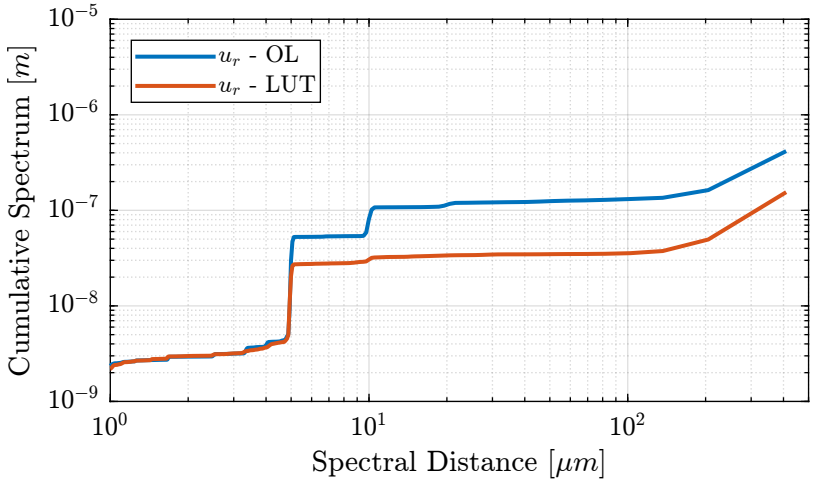

Figure 2.17: Cumulative Spectrum with and without the LUT

3 LUT creation from experimental data

In this section, the full process from measurement, filtering of data to generation of the LUT is detailed.

The computation is performed with Matlab.

3.1 Load Data

A Bragg scan is performed using `thtraj` and data are acquired using the `fast_DAQ`.

Matlab
```
%% Load Raw Data
load("scan_10_70_lut_1.mat")
```

Measured data are:

- `bragg`: Bragg angle in deg
- `dz`: distance between crystals in nm
- `dry`, `drx`: orientation errors between crystals in nrad
- `fjur`, `fjuh`, `fjd`: generated steps by the IcePAP in tens of nm

All are sampled at 10kHz with no filtering.

First, convert all the data to SI units (rad, and m).

Matlab
```
%% Convert Data to Standard Units
% Bragg Angle [rad]
bragg = pi/180*bragg;
% Rx rotation of 1st crystal w.r.t. 2nd crystal [rad]
drx   = 1e-9*drx;
% Ry rotation of 1st crystal w.r.t. 2nd crystal [rad]
dry   = 1e-9*dry;
% Z distance between crystals [m]
dz    = 1e-9*dz;
% Z error between second crystal and first crystal [m]
ddz   = 10.5e-3./(2*cos(bragg)) - dz;
% Steps for Ur motor [m]
fjur  = 1e-8*fjur;
% Steps for Uh motor [m]
fjuh  = 1e-8*fjuh;
% Steps for D motor [m]
fjd   = 1e-8*fjd;
```

3.2 IcePAP generated Steps

Here is how the steps of the IcePAP (`fjsur`, `fjsuh` and `fjsd`) are computed in mode A:

$$\begin{bmatrix}\text{fjsur}\\ \text{fjsuh}\\ \text{fjsd}\end{bmatrix}(\theta) = \text{fjs}_0 + \boldsymbol{J}_{a,111}\cdot\begin{bmatrix}0\\ \text{fjsry}\\ \text{fjsrx}\end{bmatrix} - \frac{10.5\cdot 10^{-3}}{2\cos(\theta)} \tag{3.1}$$

There is a first offset fjs_0 that is initialized once, and a second offset which is a function of `fjsry` and `fjsrx`.

Let's compute the offset which is a function of `fjsry` and `fjsrx`:

Matlab
```
fjsry = 0.53171e-3; % [rad]
fjsrx = 0.144e-3;   % [rad]

J_a_111 = [1,  0.14, -0.0675
           1,  0.14,  0.1525
           1, -0.14,  0.0425];

fjs_offset = J_a_111*[0; fjsry; fjsrx]; % ur,uh,d offsets [m]
```

6.4719e-05
9.6399e-05
-6.8319e-05

Let's now compute fjs_0 using first second of data where there is no movement and bragg axis is fixed at θ_0:

$$\text{fjs}_0 = \begin{bmatrix}\text{fjsur}\\ \text{fjsuh}\\ \text{fjsd}\end{bmatrix}(\theta_0) + \frac{10.5\cdot 10^{-3}}{2\cos(\theta_0)} - \boldsymbol{J}_{a,111}\cdot\begin{bmatrix}0\\ \text{fjsry}\\ \text{fjsrx}\end{bmatrix} \tag{3.2}$$

Matlab
```
FJ0 = ...
    mean([fjur(time < 1), fjuh(time < 1), fjd(time < 1)])' ...
    + ones(3,1)*10.5e-3./(2*cos(mean(bragg(time < 1)))) ...
    - fjs_offset; % [m]
```

0.030427
0.030427
0.030427

Values are very close for all three axis. Therefore we take the mean of the three values for fjs_0.

Matlab
```
FJ0 = mean(FJ0);
```

This approximately corresponds to the distance between the crystals for a Bragg angle of 80 degrees:

Matlab
```
10.5e-3/(2*cos(80*pi/180))
```

Results
```
0.030234
```

The measured IcePAP steps are compared with the theoretical formulas in Figure 3.1.

If we were to zoom a lot, we would see a small delay between the estimation and the steps sent by the IcePAP. This is due to the fact that the estimation is performed based on the measured Bragg angle while the IcePAP steps are based on the "requested" Bragg angle. As will be shown in the next section, there is a small delay between the requested and obtained bragg angle which explains this delay.

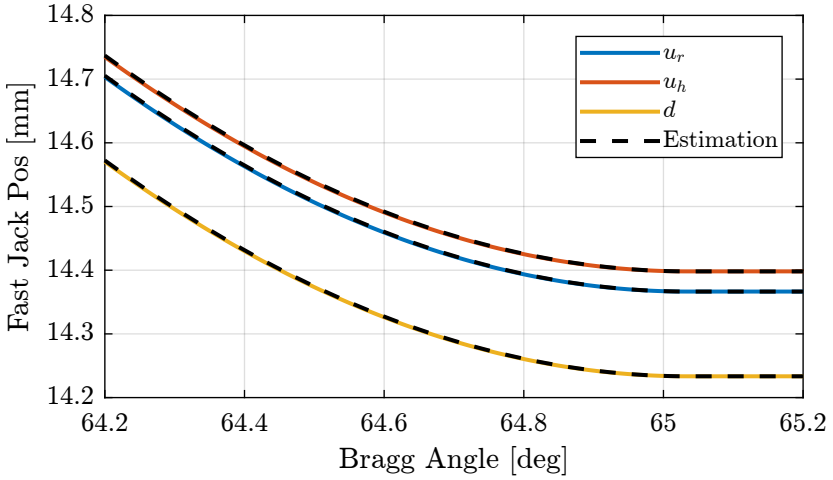

Figure 3.1: Measured IcePAP Steps and estimation from theoretical formula

3.3 Bragg and Fast Jack Velocities

In order to estimate velocities from measured positions, a filter is used which approximate a pure derivative filter.

Matlab
```
%% Filter to compute velocities
G_diff = (s/2/pi/10)/(1 + s/2/pi/10);
% Make sure the gain w = 2pi is equal to 2pi
G_diff = 2*pi*G_diff/(abs(evalfr(G_diff, 1j*2*pi)));
```

Only the high frequency amplitude is reduced to not amplified the measurement noise (Figure 3.2).

Using the filter, the Bragg velocity is estimated (Figure 3.3).

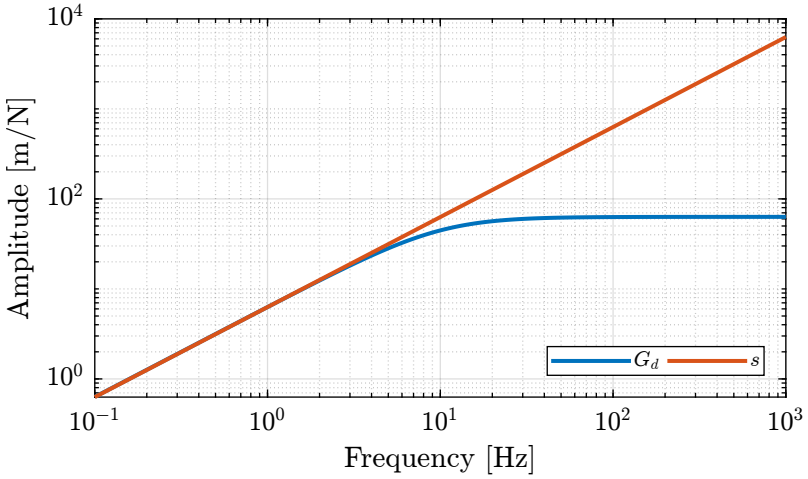

Figure 3.2: Magnitude of filter used to approximate the derivative

Matlab
```
%% Bragg Velocity
bragg_vel = lsim(G_diff, bragg, time);
```

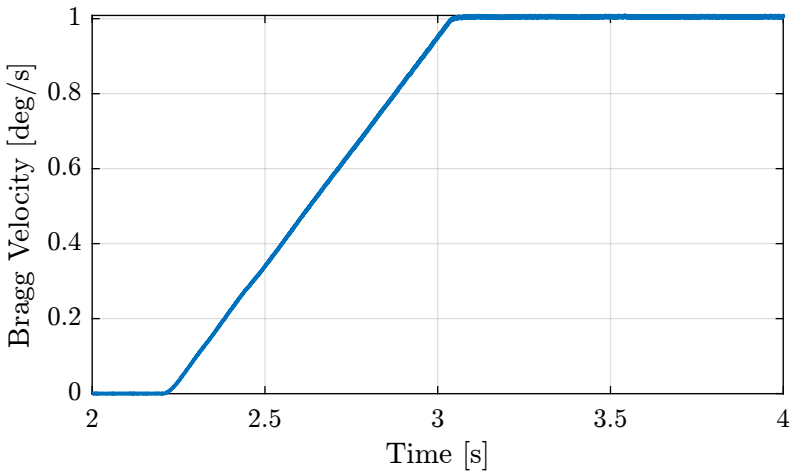

Figure 3.3: Estimated Bragg Velocity curing acceleration phase

Now, the Fast Jack velocity is estimated (Figure 3.4).

Matlab
```
%% Fast Jack Velocity
fjur_vel = lsim(G_diff, fjur, time);
fjuh_vel = lsim(G_diff, fjuh, time);
fjd_vel  = lsim(G_diff, fjd , time);
```

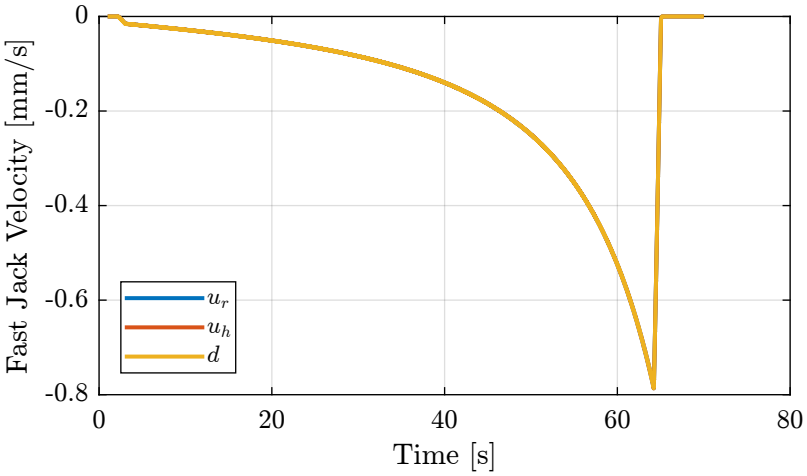

Figure 3.4: Estimated velocity of fast jacks

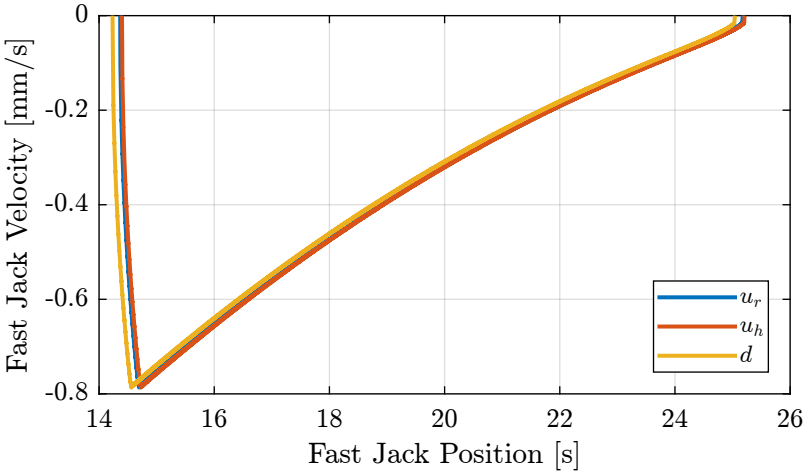

Figure 3.5: Fast Jack Velocity as a function of its position

3.4 Bragg Angle Errors / Delays

From the measured `fjur` steps generated by the IcePAP, we can estimate the steps generated corresponding to the Bragg angle.

Matlab
```
%% Estimated Bragg angle requested by IcePAP
bragg_icepap = acos(10.5e-3./(2*(FJ0 + fjs_offset(1) - fjur)));
```

The generated steps by the IcePAP and the measured angle are compared in Figure 3.6. There is clearly a lag of the Bragg angle compared to the generated IcePAP steps.

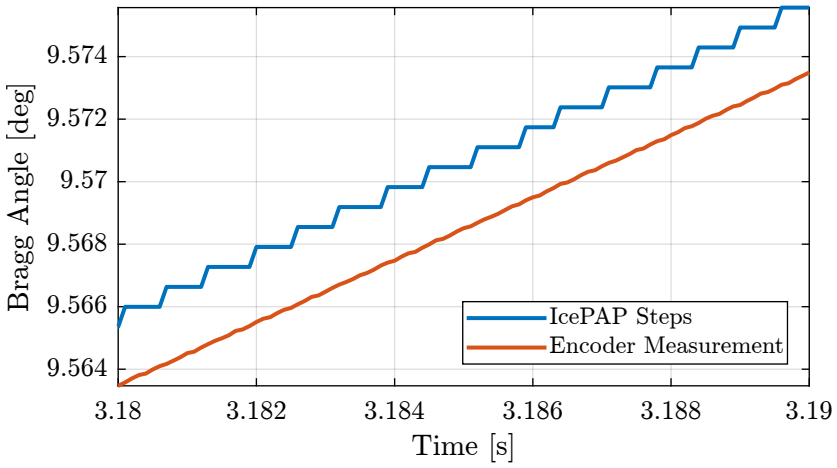

Figure 3.6: Estimated generated steps by the IcePAP and measured Bragg angle

Matlab
```
% Take only deceleration portion
scan_i = time > 60 & time < 65.1;
```

If we plot the error between the measured and the requested bragg angle as a function of the bragg velocity (Figure 3.7), we can see an almost linear relationship.

This corresponds to a "time lag" of approximately:

Results
```
2.4 ms
```

Important

There is a "lag" between the Bragg steps sent by the IcePAP and the measured angle by the encoders. This is probably due to the single integrator in the "Aerotech" controller. Indeed, two integrators are required to have no tracking error during ramp reference signals.

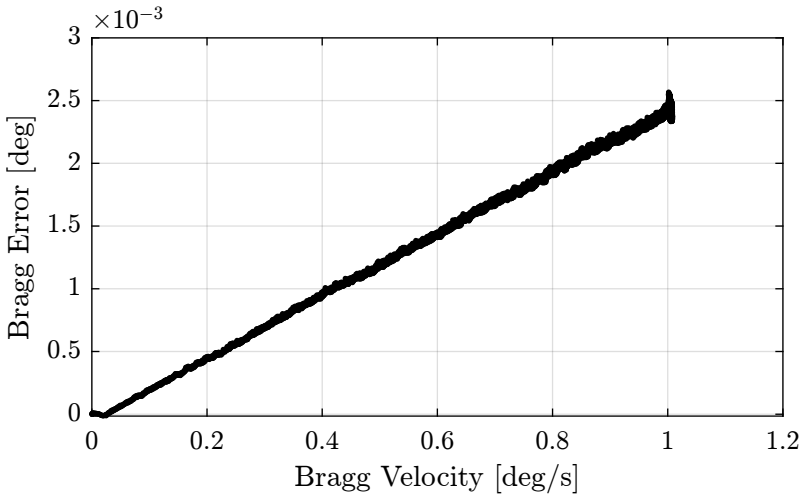

Figure 3.7: Bragg Error as a function fo the Bragg Velocity

3.5 Errors in the Frame of the Crystals

The `dz`, `dry` and `drx` measured relative motion of the crystals are defined as follows:

- An increase of `dz` means the crystals are moving away from each other
- An positive `dry` means the second crystals has positive rotation around `y`
- An positive `drx` means the second crystals has positive rotation around `x`

The error in crystals' distance `ddz` is defined as:

$$ddz(\theta) = \frac{10.5 \cdot 10^{-3}}{2\cos(\theta)} - dz(\theta) \tag{3.3}$$

Therefore, a positive `ddz` means that the second crystal is too high (fast jacks have to move down).

The errors measured in the frame of the crystals are shown in Figure 3.8.

3.6 Errors in the Frame of the Fast Jacks

From `ddz,dry,drx`, the motion errors of the jast-jacks (`fjur_e`, `fjuh_e` and `jfd_e`) as measured by the interferometers are computed.

Matlab

```
%% Actuator Jacobian
J_a_111 = [1,  0.14, -0.0675
           1,  0.14,  0.1525
           1, -0.14,  0.0425];

%% Computation of the position of the FJ as measured by the interferometers
error = J_a_111 * [ddz, dry, drx]';
```

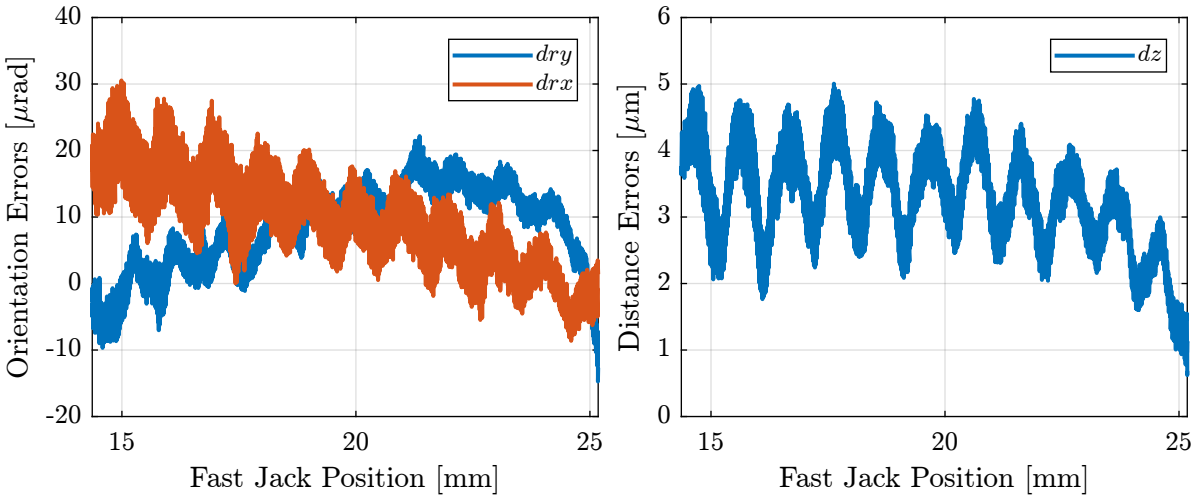

Figure 3.8: Measured errors in the frame of the crystals as a function of the fast jack position

```
fjur_e = error(1,:)'; % [m]
fjuh_e = error(2,:)'; % [m]
fjd_e  = error(3,:)'; % [m]
```

The result is shown in Figure 3.9.

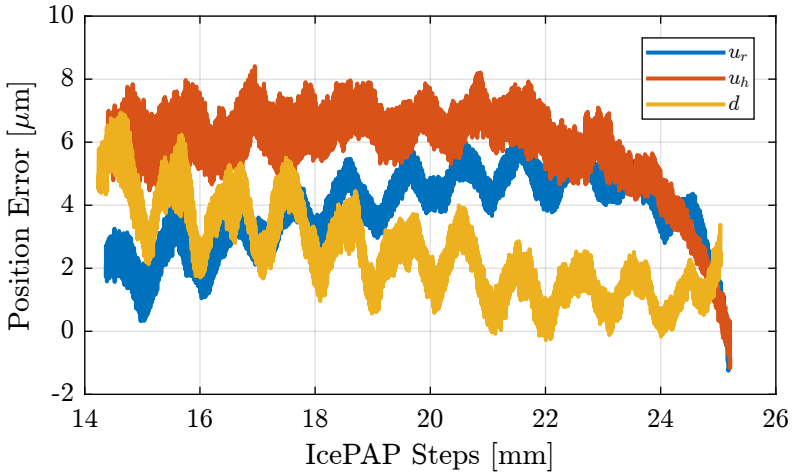

Figure 3.9: Position error of the Fast jacks

3.7 Analysis of the obtained error

The measured position of the fast jacks are displayed as a function of the IcePAP steps (Figure 3.11).

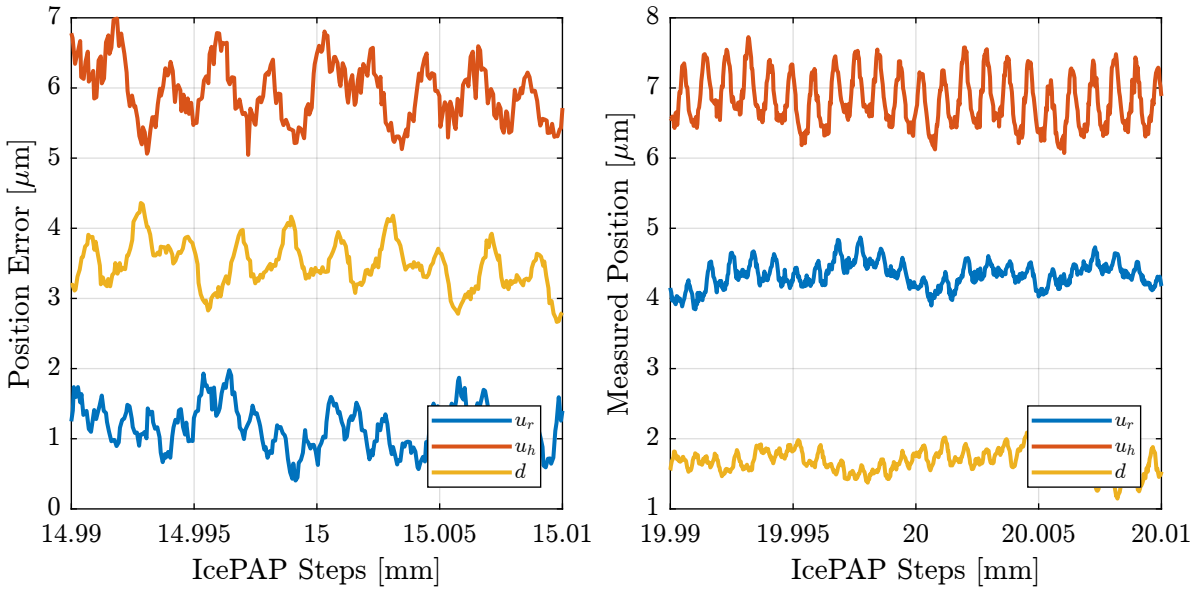

Figure 3.10: Position error of the Fast jacks - Zoom near two positions

Important

From Figure 3.11, it seems the position as a function of the IcePAP steps is not a bijection function. Therefore, a measured position can corresponds to several IcePAP Steps. This is very problematic for building a LUT that will be used to compensated the measured errors.

Also, it seems that the (spatial) period of the error depends on the actual position of the Fast Jack (and therefore of its velocity). If we compute the equivalent temporal period, we find a frequency of around 370 Hz.

In order to better investigate what is going on, a spectrogram is computed (Figure 3.12).

We clearly observe:

- Some rather constant vibrations with a frequency at around 363Hz and 374Hz. This corresponds to the clear periods in Figure 3.11. These are due to the `mcoil` stepper motor (magnetic period).
- Several frequencies which are increasing with time. These corresponds to (spatial) periodic errors of the stepper motor. The frequency of these errors are increasing because the velocity of the fast jack is also increasing with time (see Figure 3.4). The black dashed line in Figure 3.12 shows the frequency of errors with a period of $5\,\mu m$. We can also see lower frequencies corresponding to periods of $10\,\mu m$ and $20\,\mu m$ and lots of higher frequencies with are also exciting resonances of the system (second crystal) at around 200Hz

Important

As we would like to only measure the repeatable mechanical errors of the fast jacks and not the vibrations of natural frequencies of the system, we have to filter the data.

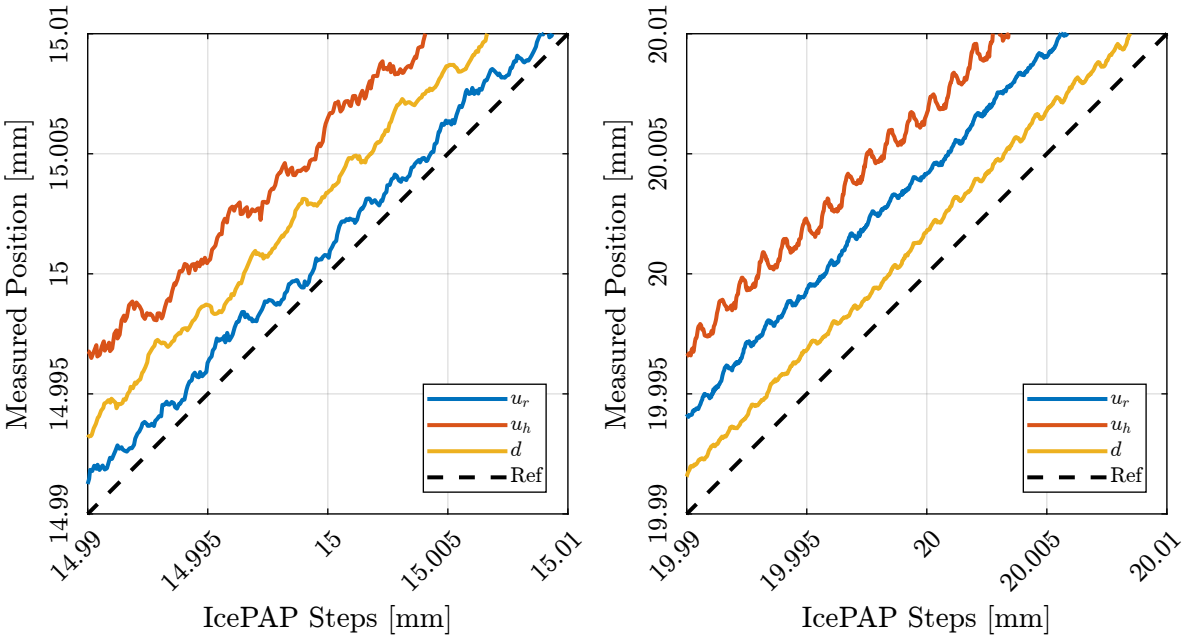

Figure 3.11: Measured Fast Jack position as a function of the IcePAP steps

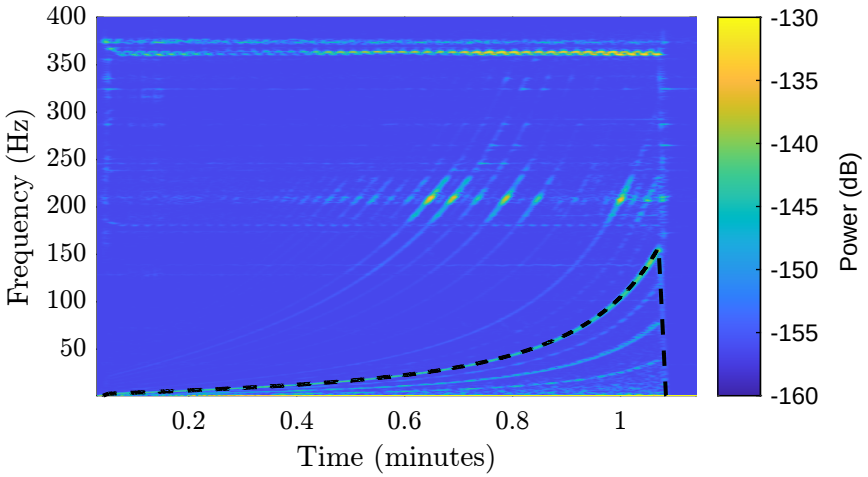

Figure 3.12: Spectrogram of the u_h errors. The black dashed line corresponds to an error with a period of $5\,\mu m$

3.8 Filtering of Data

As seen in Figure 3.12, the errors we wish to calibrate are below 160Hz while the vibrations we wish to ignore are above 200Hz. We have to use a low pass filter that does not affects frequencies below 160Hz while having good rejection above 200Hz.

The filter used for current LUT is a moving average filter with a length of 100:

Matlab
```
%% Moving Average Filter
B_mov_avg = 1/101*ones(101,1); % FIR Filter coeficients
```

We may also try a second order low pass filter:

Matlab
```
%% 2nd order Low Pass Filter
G_lpf = 1/(1 + 2*s/(2*pi*80) + s^2/(2*pi*80)^2);
```

And a FIR filter with linear phase:

Matlab
```
%% FIR with Linear Phase
Fs = 1e4; % Sampling Frequency [Hz]
B_fir = firls(1000, ... % Filter's order
              [0 140/(Fs/2) 180/(Fs/2) 1], ... % Frequencies [Hz]
              [1 1          0          0]); % Wanted Magnitudes
```

Filters' responses are computed and compared in the Bode plot of Figure 3.13.

Matlab
```
%% Computation of filters' responses
[h_mov_avg, f] = freqz(B_mov_avg, 1, 10000, Fs);
[h_fir,     ~] = freqz(B_fir,     1, 10000, Fs);
h_lpf          = squeeze(freqresp(G_lpf,  f, 'Hz'));
```

Clearly, the currently used moving average filter is filtering too much below 160Hz and too little above 200Hz. The FIR filter seems more suited for this case.

Let's now compare the filtered data.

Matlab
```
fjur_e_cur = filter(B_mov_avg, 1, fjur_e);
fjur_e_fir = filter(B_fir,     1, fjur_e);
fjur_e_lpf = lsim(G_lpf, fjur_e, time);
```

As the FIR filter introduce some delays, we can identify this relay and shift the filtered data:

Matlab
```
%% Compensate the FIR delay
delay = mean(grpdelay(B_fir));
```

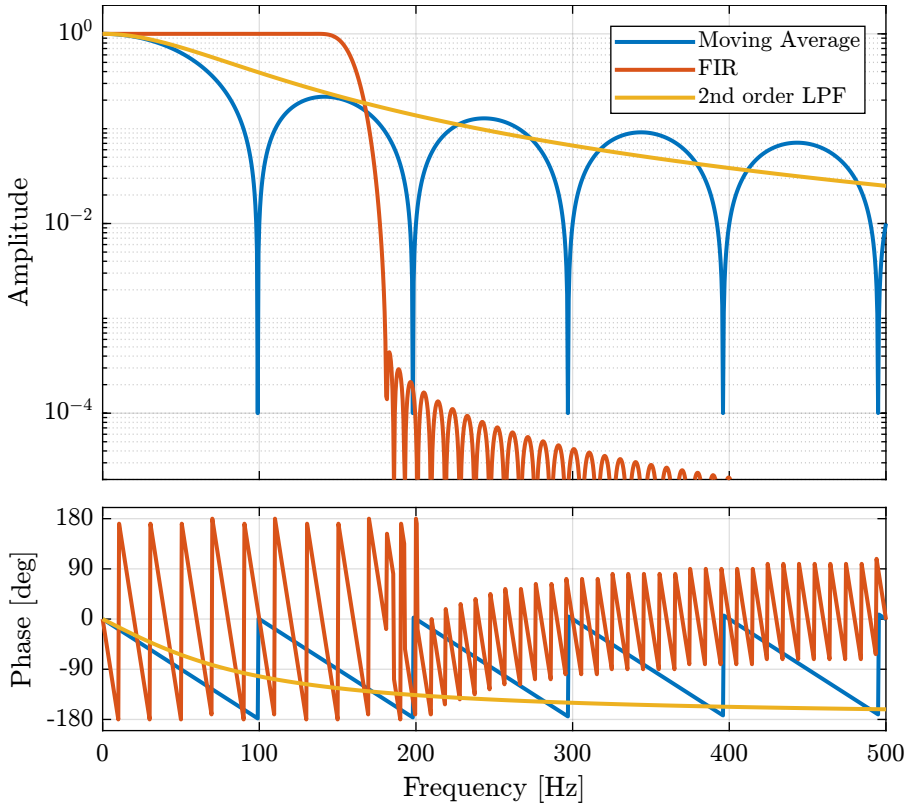

Figure 3.13: Bode plot of filters that could be used before making the LUT

Results
```
500
```

Matlab
```
fjur_e_fir(1:end-delay) = fjur_e_fir(delay+1:end);
```

The same is done for the moving average filter

Matlab
```
%% Compensate the Moving average delay
delay = mean(grpdelay(B_mov_avg));
fjur_e_cur(1:end-delay) = fjur_e_cur(delay+1:end);
```

The raw and filtered motion errors are displayed in Figure 3.14.

Important

It is shown that while the moving average average filter is working relatively well for low speeds (at around 20mm) it is not for high speeds (near 15mm). This is because the frequency of the error is above 100Hz and the moving average is flipping the sign of the filtered data.
The IIR low pass filter has some phase issues.
Finally the FIR filter is perfectly in phase while showing good attenuation of the disturbances.

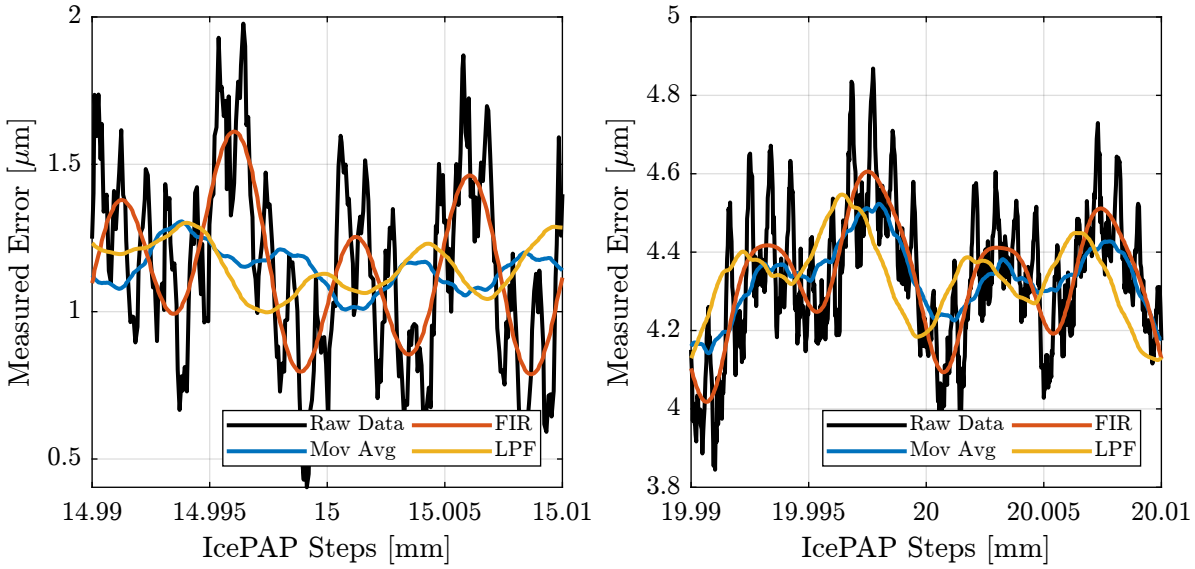

Figure 3.14: Raw measured error and filtered data

If we now look at the measured position as a function of the IcePAP steps (Figure 3.15), we can see that we obtain a monotonous function for the FIR filtered data which is great to make the LUT.

If we subtract the raw data with the FIR filtered data, we obtain the remaining motion shown in Figure 3.16 that only contains the high frequency motion not filtered.

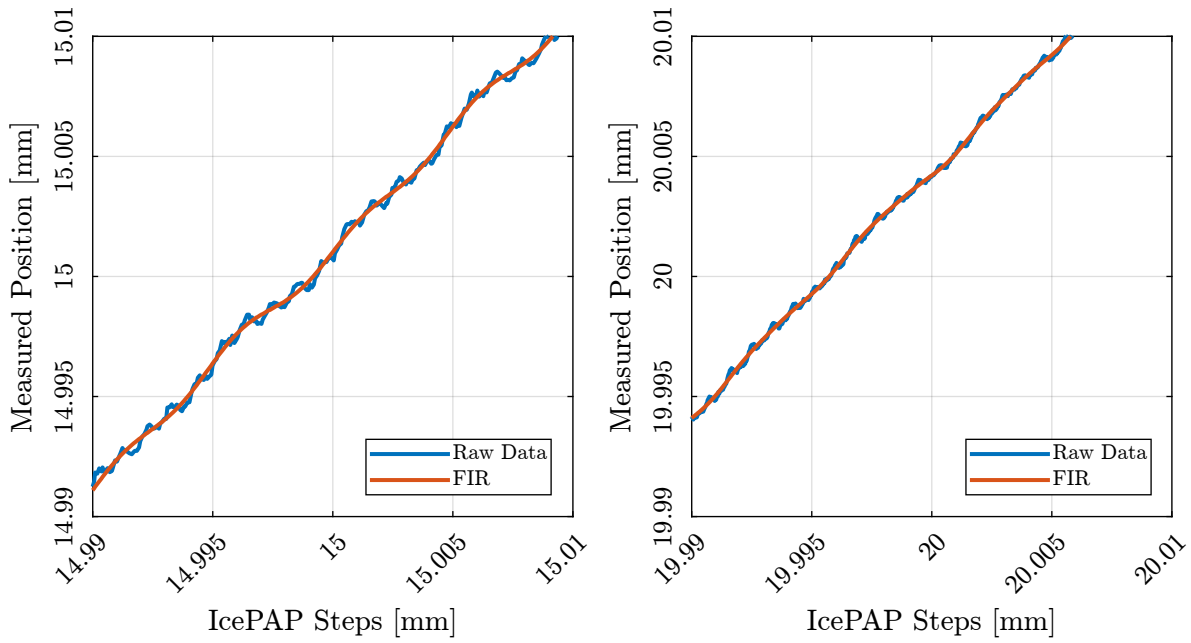

Figure 3.15: Raw measured motion and filtered motion as a function of the IcePAP Steps

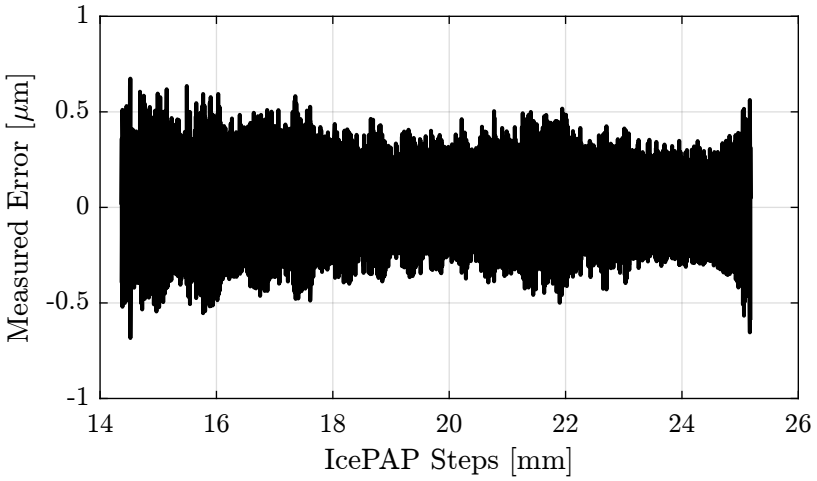

Figure 3.16: Remaining motion error after removing the filtered part

3.9 LUT creation

The procedure used to make the Lookup Table is schematically represented in Figure 3.17.

For each IcePAP step separated by a constant value (typically 1 μm) a point of the LUT is computed:

- Points where the measured position is close to the wanted ideal position (i.e. the current IcePAP step) are found
- The corresponding IcePAP step at which the Fast Jack is at the wanted position is stored in the LUT

Therefore the LUT gives the IcePAP step for which the fast jack is at the wanted position as measured by the metrology, which is what we want.

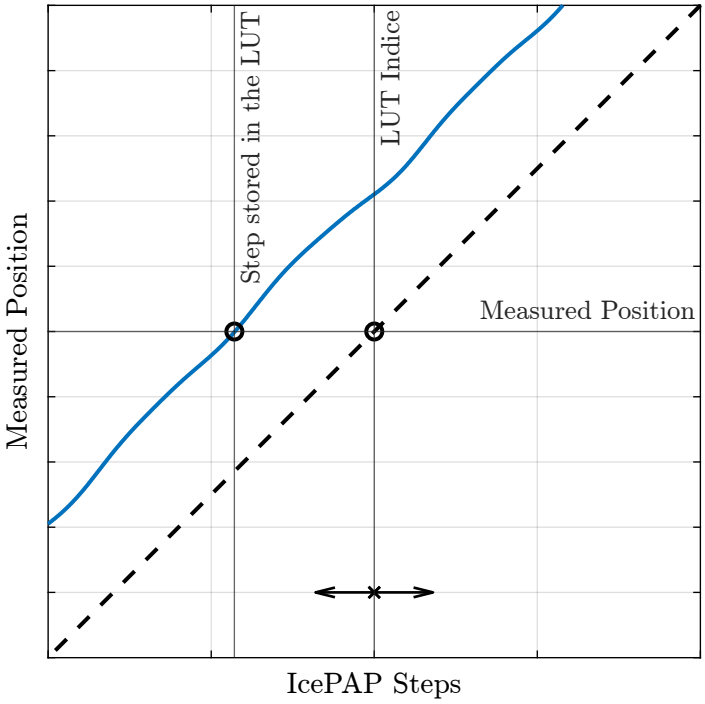

Figure 3.17: Schematic of the principle used to make the Lookup Table

Let's first initialize the LUT which is table with 4 columns and 26001 rows. The columns are:

1. IcePAP Step indices from 0 to 26mm with a step of 1 μm (thus the 26001 rows)
2. IcePAP step for `fjur` at which point the fast jack is at the wanted position
3. Same for `fjuh`
4. Same for `fjd`

All the units of the LUT are in mm. We will work in meters and convert to mm at the end.

Let's initialize the Lookup table:

Matlab
```
%% Initialization of the LUT
lut = [0:1e-6:26e-3]'*ones(1,4);
```

And verify that it has the wanted size:

Results
```
size(lut)
ans =
       26001           4
```

The measured Fast Jack position are filtered using the FIR filter:

Matlab
```
%% FIR Filter
Fs = 1e4; % Sampling Frequency [Hz]
fir_order = 1000; % Filter's order
delay = fir_order/2; % Delay induced by the filter
B_fir = firls(fir_order, ... % Filter's order
              [0 140/(Fs/2) 180/(Fs/2) 1], ... % Frequencies [Hz]
              [1 1          0          0]); % Wanted Magnitudes

%% Filtering all measured Fast Jack Position using the FIR filter
fjur_e_filt = filter(B_fir, 1, fjur_e);
fjuh_e_filt = filter(B_fir, 1, fjuh_e);
fjd_e_filt  = filter(B_fir, 1, fjd_e);

%% Compensation of the delay introduced by the FIR filter
fjur_e_filt(1:end-delay) = fjur_e_filt(delay+1:end);
fjuh_e_filt(1:end-delay) = fjuh_e_filt(delay+1:end);
fjd_e_filt( 1:end-delay) = fjd_e_filt( delay+1:end);
```

The indices where the LUT will be populated are initialized.

Matlab
```
%% Vector of Fast Jack positions [unit of lut_inc]
fjur_pos = floor(min(1e6*fjur)):floor(max(1e6*fjur));
fjuh_pos = floor(min(1e6*fjuh)):floor(max(1e6*fjuh));
fjd_pos  = floor(min(1e6*fjd )):floor(max(1e6*fjd ));
```

And the LUT is computed and shown in Figure 3.18.

Matlab
```
%% Build the LUT
for i = fjur_pos
    % Find indices where measured motion is close to the wanted one
    indices = fjur + fjur_e_filt > lut(i,1) - 500e-9 & ...
              fjur + fjur_e_filt < lut(i,1) + 500e-9;
    % Poputate the LUT with the mean of the IcePAP steps
    lut(i,2) = mean(fjur(indices));
end

for i = fjuh_pos
    % Find indices where measuhed motion is close to the wanted one
    indices = fjuh + fjuh_e_filt > lut(i,1) - 500e-9 & ...
              fjuh + fjuh_e_filt < lut(i,1) + 500e-9;
    % Poputate the LUT with the mean of the IcePAP steps
    lut(i,3) = mean(fjuh(indices));
end
```

```
for i = fjd_pos
    % Poputate the LUT with the mean of the IcePAP steps
    indices = fjd + fjd_e_filt > lut(i,1) - 500e-9 & ...
              fjd + fjd_e_filt < lut(i,1) + 500e-9;
    % Poputate the LUT
    lut(i,4) = mean(fjd(indices));
end
```

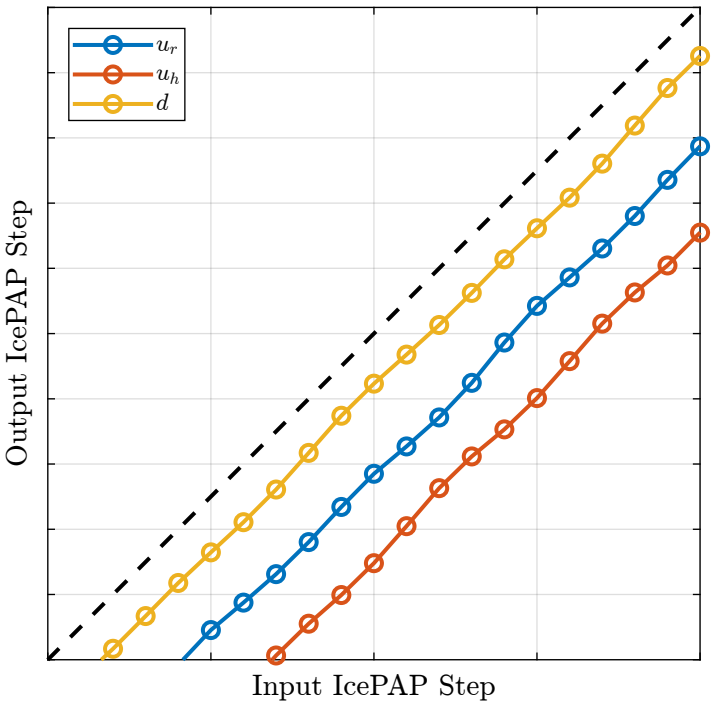

Figure 3.18: Lookup Table correction

3.10 Cubic Interpolation of the LUT

Once the LUT is built and loaded to the IcePAP, generated steps are taking the step values in the LUT and cubic spline interpolation is performed.

Matlab
```
%% Estimation of the IcePAP output steps after interpolation
fjur_out_steps = spline(lut(:,1), lut(:,2), fjur);
```

The LUT data points as well as the spline interpolation values and the ideal values are compared in Figure 3.19. It is shown that the spline interpolation seems to be quite accurate.

The difference between the perfect step generation and the step generated after spline interpolation is shown in Figure 3.20. The remaining position error is in the order of 100nm peak to peak which is acceptable here.

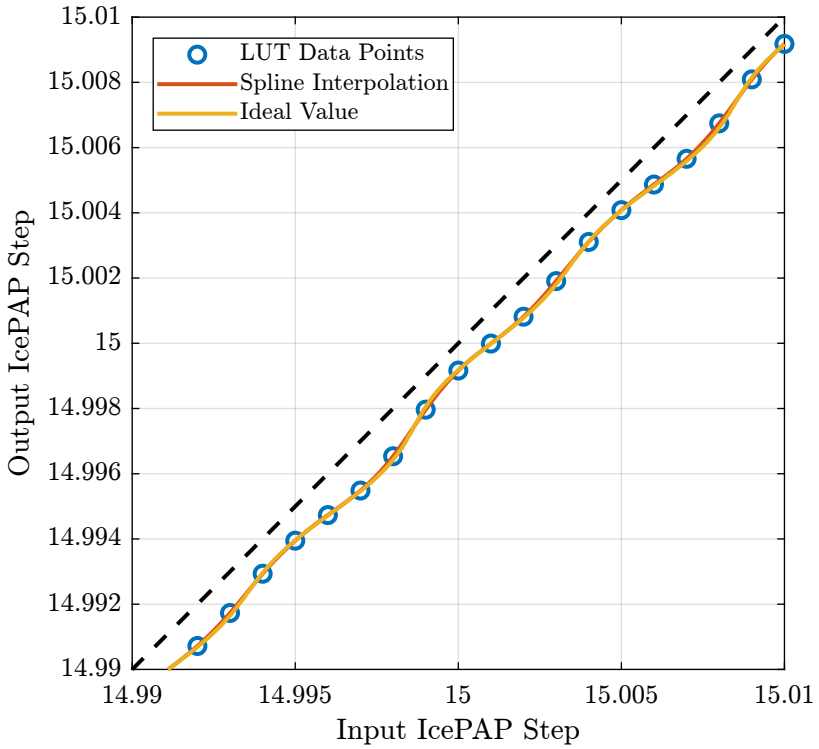

Figure 3.19: Output IcePAP Steps avec spline interpolation compared with the ideal steps

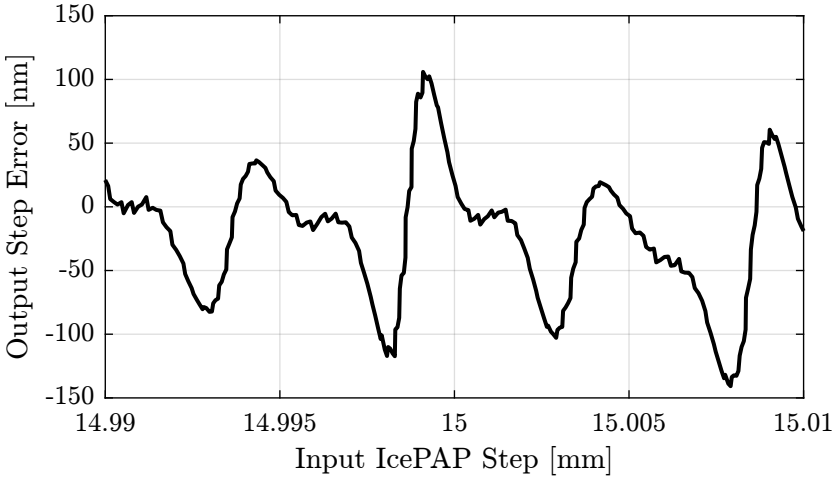

Figure 3.20: Errors on the computed IcePAP output steps after LUT generation and spline interpolation

4 Test Matlab LUT

4.1 LUT Creation

A `thtraj` scan from 10 to 70 deg is performed.

Matlab
```
%% Extract measurement Data make from BLISS
data_ol = extractDatData("/home/thomas/mnt/data_id21/21Nov/blc13420/id21/LUT_Matlab/lut_matlab_22122021_1610.dat", ...
                    {"bragg", "dz", "dry", "drx", "fjur", "fjuh", "fjd"}, ...
                    ones(7,1));
data_ol.time = 1e-4*[1:1:length(data_ol.bragg)];
save("/tmp/data_lut.mat", "-struct", "data_ol");
```

A LUT is generated from this Data.

Matlab
```
%% Generate LUT
createLUT("/tmp/data_lut.mat", "lut_matlab_22122021_1610_10_70_table.dat");
```

Matlab
```
%% Load the generated LUT
data_lut = importdata("lut_matlab_22122021_1610_10_70_table.dat");
```

The generated LUT is shown in Figure 4.1.

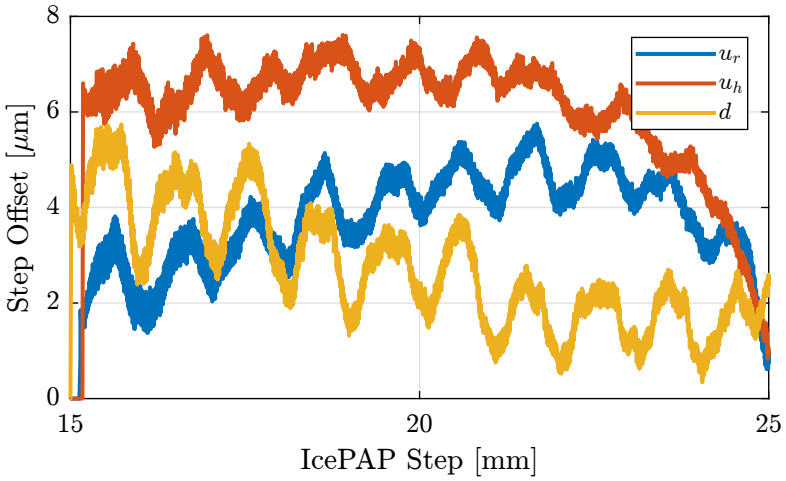

Figure 4.1: Generated LUT

4.2 Compare Mode A and Mode B

The LUT is loaded into BLISS and a new scan in mode B is performed.

Matlab

```
%% Load mode B scan data
data_B = extractDatData("/home/thomas/mnt/data_id21/21Nov/blc13420/id21/LUT_Matlab/lut_matlab_result_22122021_1616.dat", ...
                    {"bragg", "dz", "dry", "drx", "fjur", "fjuh", "fjd"}, ...
                    ones(7,1));
data_B.time = 1e-4*[1:1:length(data_B.bragg)];
save("/tmp/data_lut.mat", "-struct", "data_ol");
```

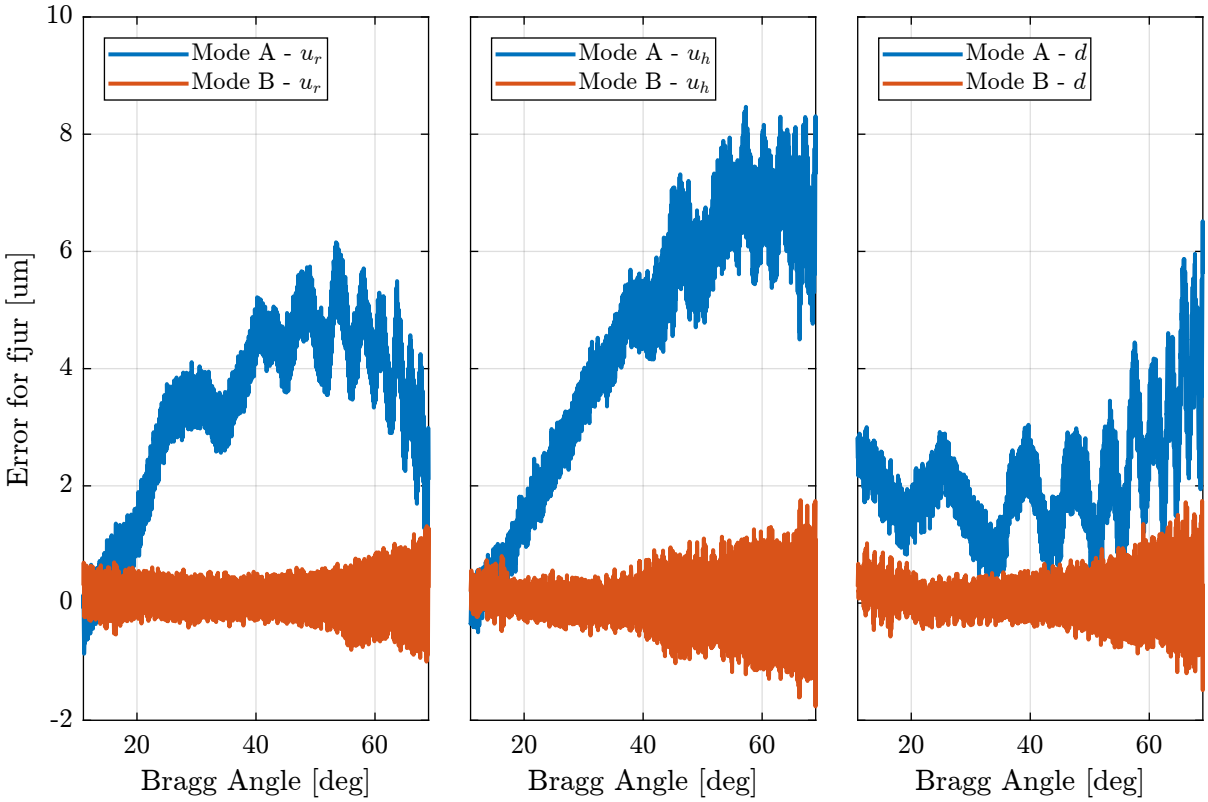

Figure 4.2: Comparison of the Raw measurement of fast jack motion errors for mode A and mode B

Now look at the low frequency motion by using a low pass filter

Matlab

```
%% FIR Filter
Fs = 1e4; % Sampling Frequency [Hz]
fir_order = 1000; % Filter's order
delay = fir_order/2; % Delay induced by the filter
B_fir = firls(fir_order, ... % Filter's order
              [0 5/(Fs/2)   10/(Fs/2) 1], ... % Frequencies [Hz]
              [1 1          0         0]); % Wanted Magnitudes

%% Filtering all measured Fast Jack Position using the FIR filter
data_B_fjur_f = filter(B_fir, 1, data_B_fjur_e);
data_B_fjuh_f = filter(B_fir, 1, data_B_fjuh_e);
data_B_fjd_f  = filter(B_fir, 1, data_B_fjd_e);

data_ol_fjur_f = filter(B_fir, 1, data_ol_fjur_e);
data_ol_fjuh_f = filter(B_fir, 1, data_ol_fjuh_e);
data_ol_fjd_f  = filter(B_fir, 1, data_ol_fjd_e);
```

```
%% Compensation of the delay introduced by the FIR filter
data_B_fjur_f(1:end-delay) = data_B_fjur_f(delay+1:end);
data_B_fjuh_f(1:end-delay) = data_B_fjuh_f(delay+1:end);
data_B_fjd_f( 1:end-delay) = data_B_fjd_f( delay+1:end);

data_ol_fjur_f(1:end-delay) = data_ol_fjur_f(delay+1:end);
data_ol_fjuh_f(1:end-delay) = data_ol_fjuh_f(delay+1:end);
data_ol_fjd_f( 1:end-delay) = data_ol_fjd_f( delay+1:end);
```

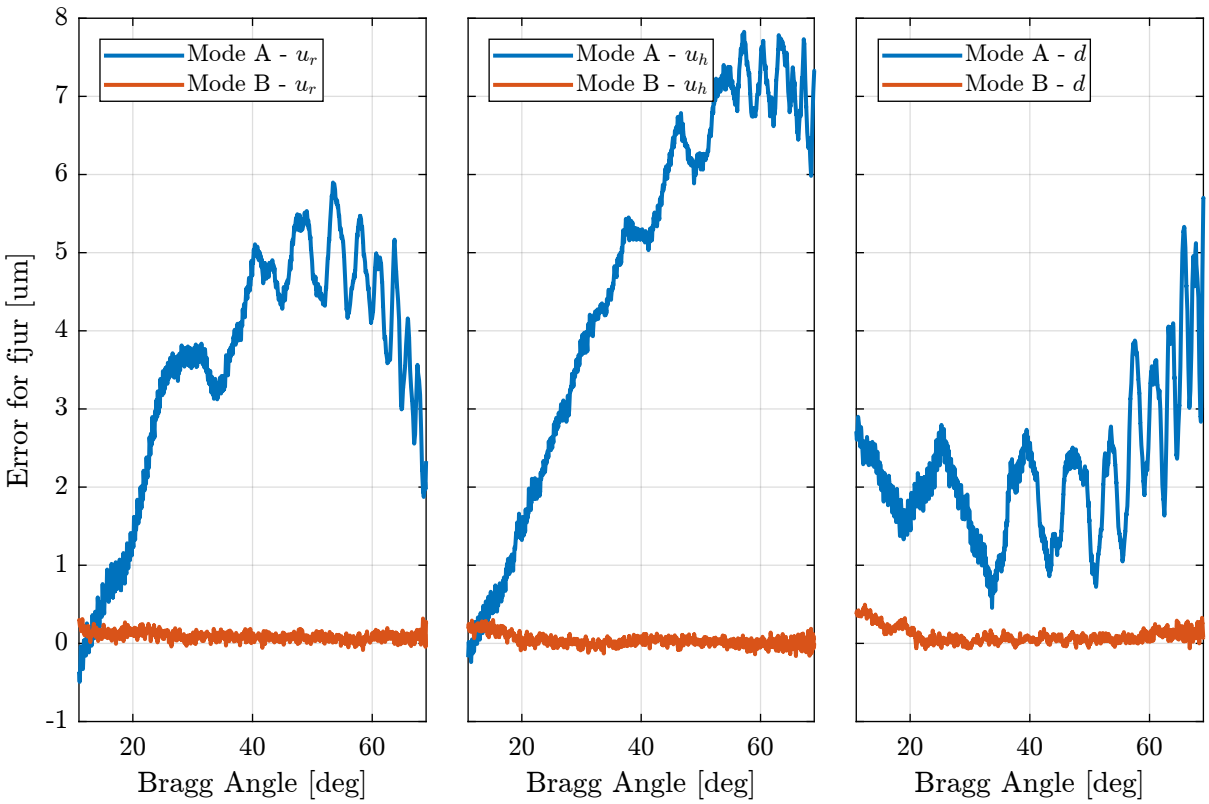

Figure 4.3: Comparison of the Raw measurement of fast jack motion errors for mode A and mode B

4.3 Check IcePAP Steps with LUT

Compare the theoretical steps and the measured steps when the LUT is on.

Estimate wanted `fjur` steps.

Matlab

```
FJ0 = 0.030427;

fjsry = 0.53171e-3; % [rad]
fjsrx = 0.144e-3;   % [rad]

J_a_111 = [1,  0.14, -0.0675
           1,  0.14,  0.1525
           1, -0.14,  0.0425];
```

```
fjs_offset = J_a_111*[0; fjsry; fjsrx]; % ur,uh,d offsets [m]

fjur_th = 1e3*(FJ0 + fjs_offset(1) - 10.5e-3./(2*cos(pi/180*data_B.bragg)));
```

Matlab

```
%% Estimation of the IcePAP output steps after interpolation
fjur_out_steps = spline(data_lut(:,1), data_lut(:,2), fjur_th);
```

Difference between theoretical step using LUT and sent step using IcePAP

Matlab

```
figure;
plot(data_B.bragg, 1e-2*data_B.fjur - 1e3*fjur_out_steps)
xlabel('Bragg [deg]'); ylabel('Difference on fjpur [$\mu$m]');
xlim([10, 70]);
```

This difference can be explained by the fact that we are basing the theoretical step after LUT on the measured Bragg angle and not on the requested one (there is a delay).

5 LUT Without `mcoil`

5.1 Load

Matlab
```
%% Extract measurement Data make from BLISS
data_ol =
↪  extractDatData("/home/thomas/mnt/data_id21/21Nov/blc13420/id21/LUT_without_mcoil/lut_without_mcoil_06012022_1644.dat", ...
                  {"bragg", "dz", "dry", "drx", "fjur", "fjuh", "fjd"}, ...
                  ones(7,1));
data_ol.time = 1e-4*[1:1:length(data_ol.bragg)];
% save("/tmp/data_lut.mat", "-struct", "data_ol");
```

Matlab
```
%% Extract measurement Data make from BLISS
data_bis = extractDatData("/home/thomas/mnt/data_id21/21Nov/blc13420/id21/LUT_Matlab/lut_matlab_06012022_1651.dat", ...
                  {"bragg", "dz", "dry", "drx", "fjur", "fjuh", "fjd"}, ...
                  ones(7,1));
data_bis.time = 1e-4*[1:1:length(data_bis.bragg)];
% save("/tmp/data_lut.mat", "-struct", "data_ol");
```

Matlab
```
%% FIR Filter
Fs = 1e4; % Sampling Frequency [Hz]
fir_order = 1000; % Filter's order
delay = fir_order/2; % Delay induced by the filter
B_fir = firls(fir_order, ... % Filter's order
              [0 5/(Fs/2)   10/(Fs/2) 1], ... % Frequencies [Hz]
              [1 1          0         0]); % Wanted Magnitudes
```

6 Optimal Trajectory to make the LUT

In this section, the problem of generating an adequate trajectory to make the LUT is studied.

The problematic is the following:

1. the positioning errors of the fast jack should be measured
2. all external disturbances and measurement noise should be filtered out.

The main difficulty is that the frequency of both the positioning errors errors and the disturbances are a function of the scanning velocity.

First, the frequency of the disturbances as well as the errors to be measured are described and a filter is designed to optimally separate disturbances from positioning errors (Section 6). The relation between the Bragg angular velocity and fast jack velocity is studied in Section 6.2. Next, a trajectory with constant fast jack velocity (Section 6.3) and with constant Bragg angular velocity (Section 6.5) are simulated to understand their limitations. Finally, it is proposed to perform a scan in two parts (one part with constant fast jack velocity and the other part with constant bragg angle velocity) in Section 6.5.

6.1 Filtering Disturbances and Noise

Based on measurements made in mode A (without LUT or feedback control), several disturbances could be identified:

- vibrations coming from from the `mcoil` motor
- vibrations with constant frequencies at 29Hz (pump), 34Hz (air conditioning) and 45Hz (unidentified)

These disturbances as well as the noise of the interferometers should be filtered out, and only the fast jack motion errors should be left untouched.

Therefore, the goal is to make a scan such that during all the scan, the frequencies of the errors induced by the fast jack have are smaller than the frequencies of all other disturbances. Then, it is easy to use a filter to separate the disturbances and noise from the positioning errors of the fast jack.

Errors induced by the Fast Jack

The Fast Jack is composed of one stepper motor, and a planetary roller screw with a pitch of 1mm/turn. The stepper motor as 50 pairs of magnetic poles, and therefore positioning errors are to be expected every 1/50th of turn (and its harmonics: 1/100th of turn, 1/200th of turn, etc.).

One pair of magnetic pole corresponds to an axial motion of $20\,\mu m$. Therefore, errors are to be expected with a period of $20\,\mu m$ and harmonics at $10\,\mu m$, $5\,\mu m$, $2.5\,\mu m$, etc.

As the LUT has one point every $1\,\mu m$, we wish to only measure errors with a period of $20\,\mu m$, $10\,\mu m$ and $5\,\mu m$. Indeed, errors with smaller periods are small in amplitude (i.e. not worth to compensate) and are difficult to model with the limited number of points in the LUT.

The frequency corresponding to errors with a period of $5\,\mu m$ at 1mm/s is:

Results
```
Frequency or errors with period of 5um/s at 1mm/s is: 200.0 [Hz]
```

We wish that the frequency of the error corresponding to a period of $5\,\mu m$ to be smaller than the smallest disturbance to be filtered.

As the main disturbances are at 34Hz and 45Hz, we constrain the the maximum axial velocity of the Fast Jack such that the positioning error has a frequency bellow 25Hz:

Matlab
```
max_fj_vel = 25*1e-3/(1e-3/5e-6); % [m/s]
```

Results
```
Maximum Fast Jack velocity: 0.125 [mm/s]
```

Important

Therefore, the Fast Jack scans should be scanned at rather low velocity for the positioning errors to be at sufficiently low frequency.

Vibrations induced by `mcoil`

The `mcoil` system is composed of one stepper motor and a reducer such that one stepper motor turns makes the `mcoil` axis to rotate 0.2768 degrees. When scanning the `mcoil` motor, periodic vibrations can be measured by the interferometers.

It has been identified that the period of these vibrations are corresponding to the period of the magnetic poles (50 per turn as for the Fast Jack stepper motors).

Therefore, the frequency of these periodic errors are a function of the angular velocity. With an angular velocity of 1deg/s, the frequency of the vibrations are expected to be at:

Results
```
Fundamental frequency at 1deg/s: 180.6 [Hz]
```

We wish the frequency of these errors to be at minimum 34Hz (smallest frequency of other disturbances). The corresponding minimum `mcoil` velocity is:

Matlab
```
min_bragg_vel = 34/(50/0.2768); % [deg/s]
```

Results
```
Min mcoil velocity is 0.19 [deg/s]
```

Important

Regarding the `mcoil` motor, the problematic is to not scan too slowly. It should however be checked whether the amplitude of the induced vibrations is significant of not.

Note that the maximum bragg angular velocity is:

Matlab
```
max_bragg_vel = 1; % [deg/s]
```

Measurement noise of the interferometers

The motion of the fast jacks are measured by interferometers which have some measurement noise. It is wanted to filter this noise to acceptable values to have a clean measured position.

As the interferometer noise has a rather flat spectral density, it is easy to estimate its RMS value as a function of the cut-off frequency of the filter.

The RMS value of the filtered interferometer signal as a function of the cutoff frequency of the low pass filter is computed and shown in Figure 6.1.

Important

As the filter will have a cut-off frequency between 25Hz (maximum frequency of the positioning errors) and 34Hz (minimum frequency of disturbances), a filtered measurement noise of 0.1nm RMS is to be expected.

Note

Figure 6.1 is a rough estimate. Precise estimation can be done by measuring the spectral density of the interferometer noise experimentally.

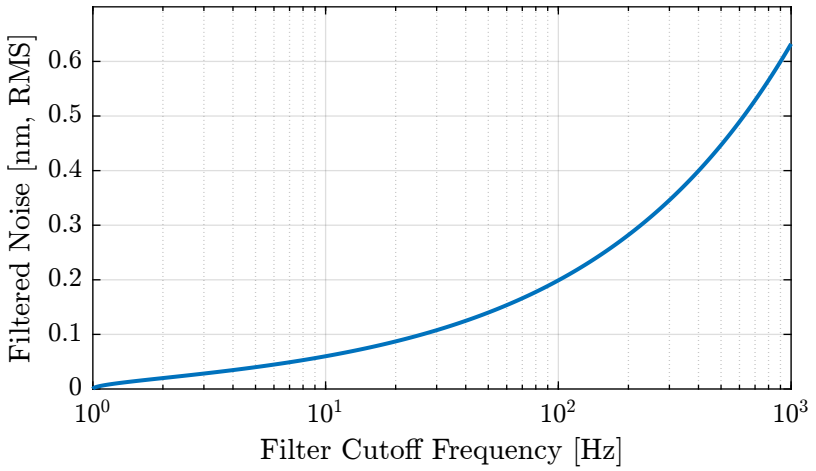

Figure 6.1: Filtered noise RMS value as a function of the low pass filter cut-off frequency

Interferometer - Periodic non-linearity

Interferometers can also show periodic non-linearity with a (fundamental) period equal to half the wavelength of its light (i.e. 765nm for Attocube) and with unacceptable amplitudes (up to tens of nanometers).

The minimum frequency associated with these errors is therefore a function of the fast jack velocity. With a velocity of 1mm/s, the frequency is:

Results

```
Fundamental frequency at 1mm/s: 1307.2 [Hz]
```

We wish these errors to be at minimum 34Hz (smallest frequency of other disturbances). The corresponding minimum velocity of the Fast Jack is:

Matlab

```
min_fj_vel = 34*1e-3/(1e-3/765e-9); % [m/s]
```

Results

```
Minimum Fast Jack velocity is 0.026 [mm/s]
```

Important

The Fast Jack Velocity should not be too low or the frequency of the periodic non-linearity of the interferometer would be too small to be filtered out (i.e. in the pass-band of the filter).

Implemented Filter

Let's now verify that it is possible to implement a filter that keep everything untouched below 25Hz and filters everything above 34Hz.

To do so, a FIR linear phase filter is designed:

Matlab
```
%% FIR with Linear Phase
Fs = 1e4; % Sampling Frequency [Hz]
B_fir = firls(5000, ... % Filter's order
              [0 25/(Fs/2) 34/(Fs/2) 1], ... % Frequencies [Hz]
              [1 1          0          0]); % Wanted Magnitudes
```

Its amplitude response is shown in Figure 6.2. It is confirmed that the errors to be measured (below 25Hz) are left untouched while the disturbances above 34Hz are reduced by at least a factor 10^4.

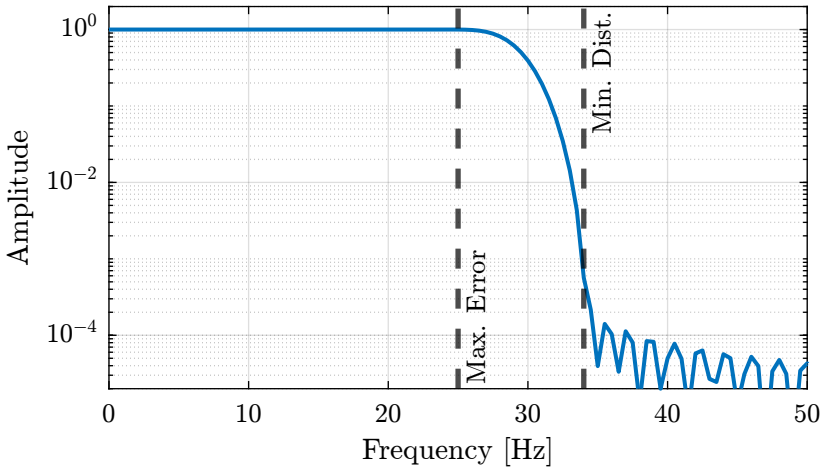

Figure 6.2: FIR filter's response

To have such a steep change in gain, the order of the filter is rather large. This has the negative effect of inducing large time delays:

Results
```
Induced time delay is 0.25 [s]
```

This time delay is only requiring us to start the acquisition 0.25 seconds before the important part of the scan is performed (i.e. the first 0.25 seconds of data cannot be filtered).

6.2 First Estimation of the optimal trajectory

Based on previous analysis (Section 6.1), minimum and maximum fast jack velocities and bragg angular velocities could be determined. These values are summarized in Table 6.1. Therefore, if during the scan the velocities are within the defined bounds, it will be very easy to filter the data and extract only the relevant information (positioning error of the fast jack).

Table 6.1: Minimum and Maximum estimated velocities

	Min	Max
Bragg Angular Velocity [deg/s]	0.188	1.0
Fast Jack Velocity [mm/s]	0.026	0.125

We now wish to see if it is possible to perform a scan from 5deg to 75deg of bragg angle while keeping the velocities within the bounds in Table 6.1.

To study that, we can compute the relation between the Bragg angular velocity and the Fast Jack velocity as a function of the Bragg angle.

To do so, we first look at the relation between the Bragg angle θ_b and the Fast Jack position d_{FJ}:

$$d_{FJ}(t) = d_0 - \frac{10.5 \cdot 10^{-3}}{2\cos\theta_b(t)} \tag{6.1}$$

with $d_0 \approx 0.030427\,m$.

Then, by taking the time derivative, we obtain the relation between the Fast Jack velocity $\dot{d}_{\mathrm{FJ}}$ and the Bragg angular velocity $\dot{\theta}_b$ as a function of the bragg angle θ_b:

$$\boxed{\dot{d}_{FJ}(t) = -\dot{\theta}_b(t) \cdot \frac{10.5 \cdot 10^{-3}}{2} \cdot \frac{\tan\theta_b(t)}{\cos\theta_b(t)}} \tag{6.2}$$

The relation between the Bragg angular velocity and the Fast Jack velocity is computed for several angles starting from 5degrees up to 75 degrees and this is shown in Figure 6.3.

Important

From Figure 6.3, it is clear that only Bragg angles from apprimately 15 to 70 degrees can be scanned by staying in the "perfect" zone (defined by the dashed black lines).
To scan smaller bragg angles, either the maximum bragg angular velocity should be increased or the minimum fast jack velocity decreased (accepting some periodic non-linearity to be measured).
To scan higher bragg angle, either the maximum fast jack velocity should be increased or the minimum bragg angular velocity decreased (taking the risk to have some disturbances from the `mcoil` motion in the signal).

For Bragg angles between 15 degrees and 70 degrees, several strategies can be chosen:

- Constant Fast Jack velocity (Figure 6.4 - Left):
  1. Go from 15 degrees to 44 degrees at minimum fast jack velocity
  2. Go from 44 degrees to 70 degrees at maximum fast jack velocity
- Constant Bragg angular velocity (Figure 6.4 - Right):
  1. Go from 15 degrees to 44 degrees at maximum angular velocity

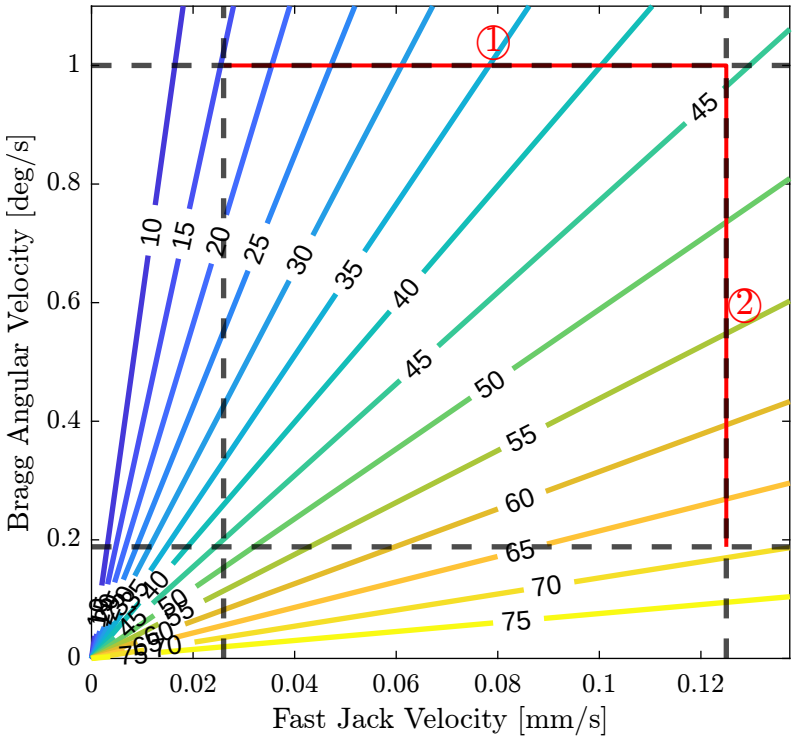

Figure 6.3: Bragg angular velocity as a function of the fast jack velocity for several bragg angles (indicated by the colorful lines in degrees). Black dashed lines indicated minimum/maximum bragg angular velocities as well as minimum/maximum fast jack velocities

2. Go from 44 to 70 degrees at minimum angular velocity

- A mixed of constant bragg angular velocity and constant fast jack velocity (Figure 6.3 - Red line)

  1. from 15 to 44 degrees with maximum Bragg angular velocity

  2. from 44 to 70 degrees with maximum Bragg angular velocity

The third option is studied in Section 6.5

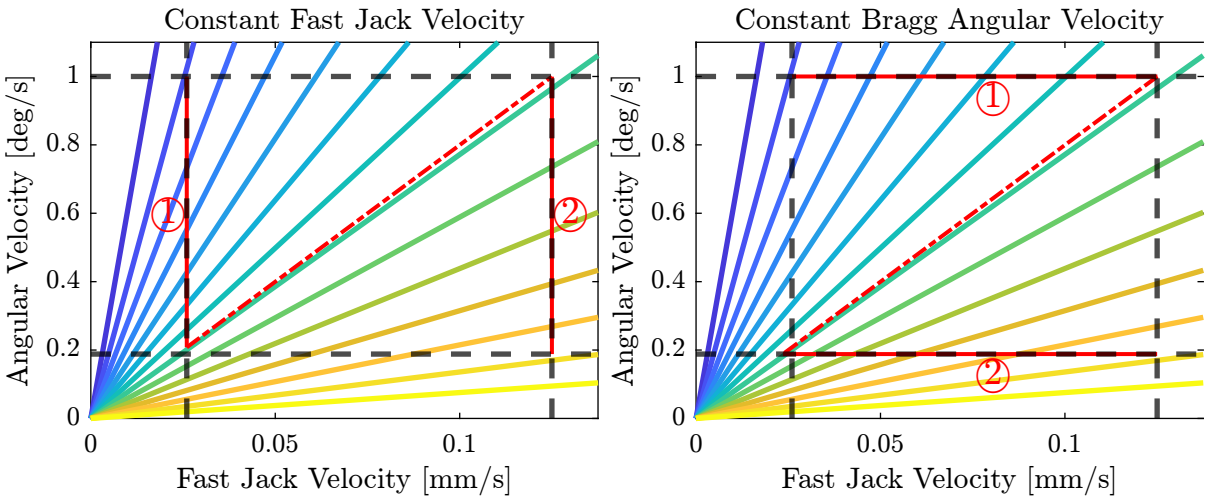

Figure 6.4: Angular velocity and fast jack velocity during two scans from 5 to 75 degrees. On the left for a scan with constant fast jack velocity. On the right for a scan with constant Bragg angular velocity.

6.3 Constant Fast Jack Velocity

In this section, a scan with constant fast jack velocity is studied.

It was shown in Section 6 that the maximum Fast Jack velocity should be 0.125mm/s in order for the frequency corresponding to the period of 5 μm to be smaller than 25Hz.

Let's generate a trajectory between 5deg and 75deg Bragg angle with constant Fast Jack velocity at 0.125mm/s.

Matlab

```
%% Compute extreme fast jack position
fj_max = 0.030427 - 10.5e-3/(2*cos(pi/180*5)); % Smallest FJ position [m]
fj_min = 0.030427 - 10.5e-3/(2*cos(pi/180*75)); % Largest FJ position [m]

%% Compute Fast Jack Trajectory
t = 0:0.1:(fj_max - fj_min)/max_fj_vel; % Time vector [s]
fj_pos = fj_max - t*max_fj_vel; % Fast Jack Position [m]

%% Compute corresponding Bragg trajectory
bragg_pos = acos(10.5e-3./(2*(0.030427 - fj_pos))); % [rad]
```

The Fast Jack position as well as the Bragg angle are shown as a function of time in Figure 6.5.

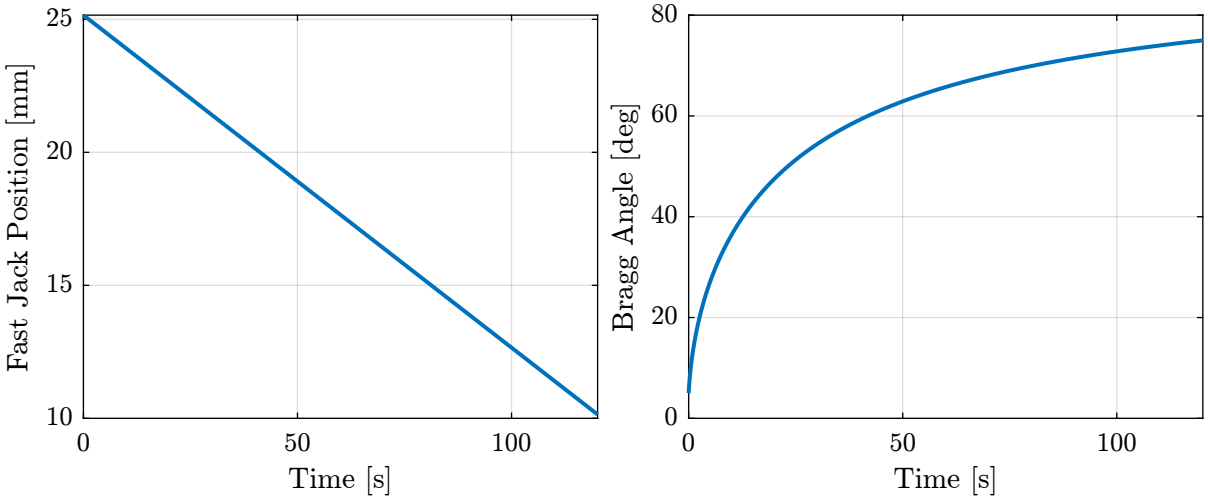

Figure 6.5: Trajectory with constant Fast Jack Velocity

Let's now compute the Bragg angular velocity for this scan (Figure 6.6). It is shown that for large Fast Jack positions / small bragg angles, the bragg angular velocity is too large.

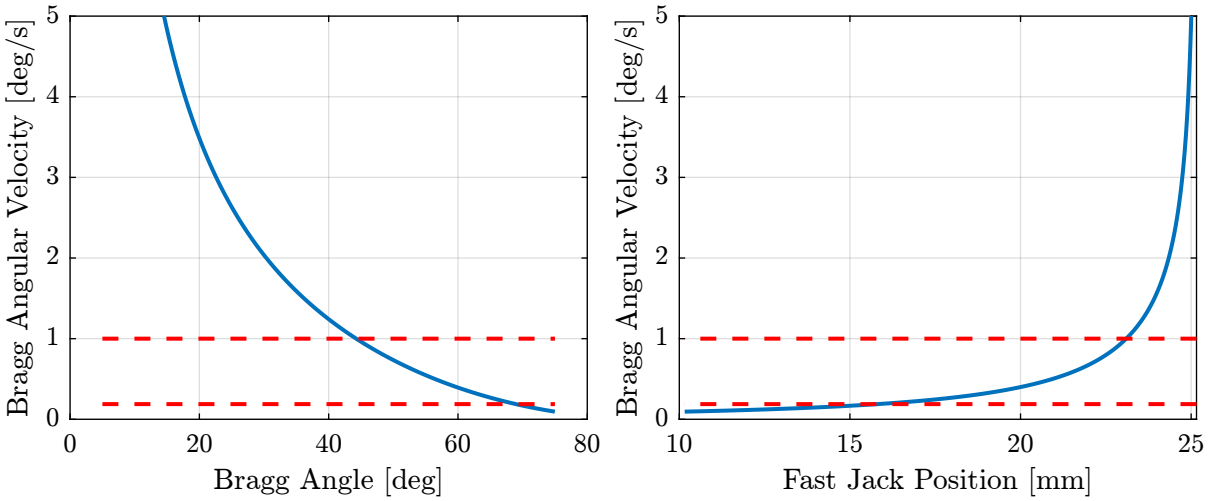

Figure 6.6: Bragg Velocity as a function of the bragg angle or fast jack position

Important

Between 45 and 70 degrees, the scan can be performed with **constant Fast Jack velocity** equal to 0.125 mm/s.

6.4 Constant Bragg Angular Velocity

Let's now study a scan with a constant Bragg angular velocity of 1deg/s.

```Matlab
%% Time vector for the Scan with constant angular velocity
t = 0:0.1:(75 - 5)/max_bragg_vel; % Time vector [s]

%% Bragg angle during the scan
bragg_pos = 5 + t*max_bragg_vel; % Bragg Angle [deg]

%% Computation of the Fast Jack Position
fj_pos = 0.030427 - 10.5e-3./(2*cos(pi/180*bragg_pos)); % FJ position [m]
```

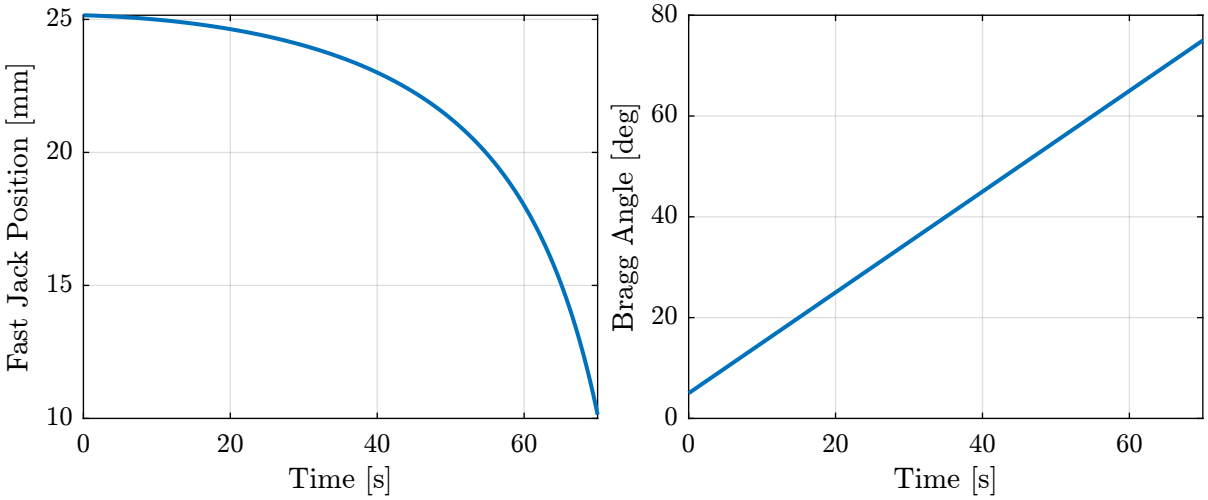

Figure 6.7: Trajectory with constant Bragg angular velocity

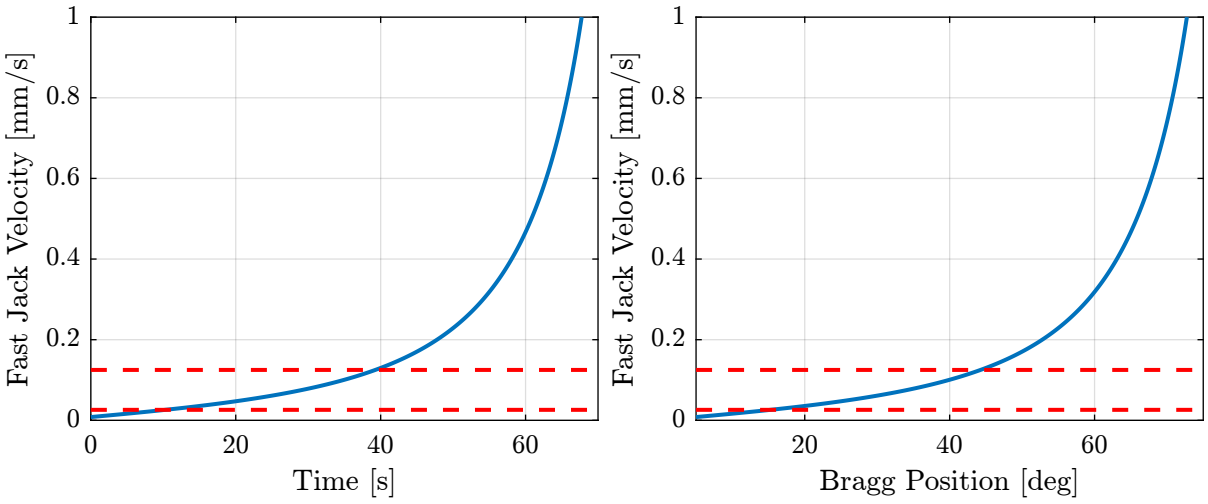

Figure 6.8: Fast Jack Velocity with a constant bragg angular velocity

Important

Between 15 and 45 degrees, the scan can be performed with a **constant Bragg angular velocity** equal to 1 deg/s.

6.5 Mixed Trajectory

Let's combine a scan with constant Bragg angular velocity for small bragg angles (< 44.3 deg) with a scan with constant Fast Jack velocity for large Bragg angle (> 44.3 deg). The scan is performed from 5 degrees to 75 degrees.

Parameters for the scan are defined below:

Matlab
```
%% Bragg Positions
bragg_start = 5; % Start Bragg angle [deg]
bragg_mid   = 44.3; % Transition between constant FJ vel and constant Bragg vel [deg]
bragg_end   = 75; % End Bragg angle [deg]

%% Fast Jack Positions
fj_start = 0.030427 - 10.5e-3/(2*cos(pi/180*bragg_start)); % Start FJ position [m]
fj_mid   = 0.030427 - 10.5e-3/(2*cos(pi/180*bragg_mid)); % Mid FJ position [m]
fj_end   = 0.030427 - 10.5e-3/(2*cos(pi/180*bragg_end)); % End FJ position [m]

%% Time vectors
Ts = 0.1; % Sampling Time [s]
t_c_bragg = 0:Ts:(bragg_mid-bragg_start)/max_bragg_vel; % Time Vector for constant bragg velocity [s]
t_c_fj = Ts+[0:Ts:(fj_mid-fj_end)/max_fj_vel]; % Time Vector for constant Fast Jack velocity [s]
```

Positions for the first part of the scan at constant Bragg angular velocity are computed:

Matlab
```
%% Constant Bragg Angular Velocity
bragg_c_bragg = bragg_start + t_c_bragg*max_bragg_vel; % [deg]
fj_c_bragg = 0.030427 - 10.5e-3./(2*cos(pi/180*bragg_c_bragg)); % FJ position [m]
```

And positions for the part of the scan with constant Fast Jack Velocity are computed:

Matlab
```
%% Constant Bragg Angular Velocity
fj_c_fj = fj_mid - t_c_fj*max_fj_vel; % FJ position [m]
bragg_c_fj = 180/pi*acos(10.5e-3./(2*(0.030427 - fj_c_fj))); % [deg]
```

Fast Jack position as well as Bragg angle are displayed as a function of time in Figure 6.9.

The Fast Jack velocity as well as the Bragg angular velocity are shown as a function of the Bragg angle in Figure 6.10.

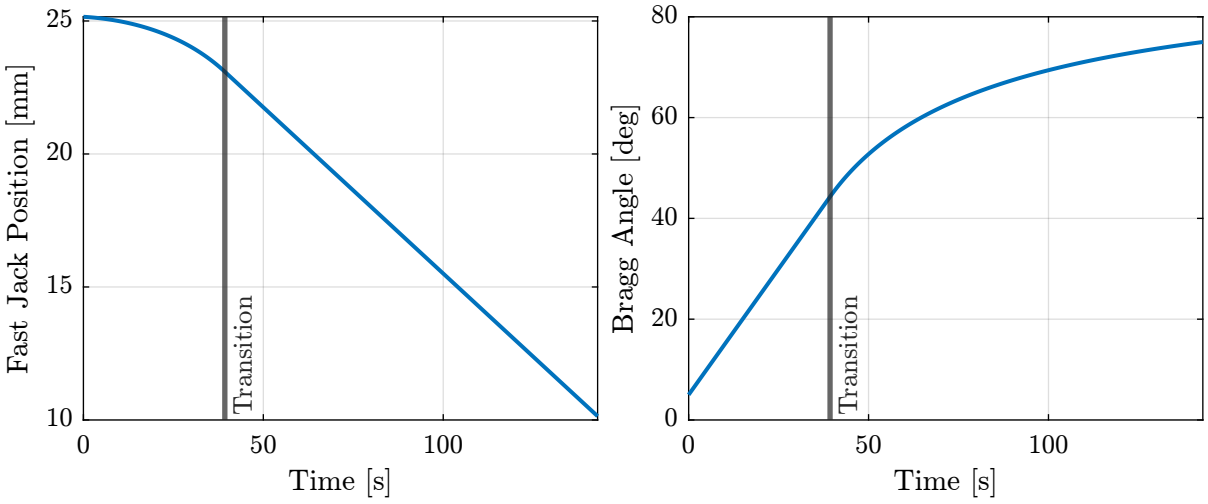

Figure 6.9: Fast jack trajectories and Bragg angular velocity during the scan

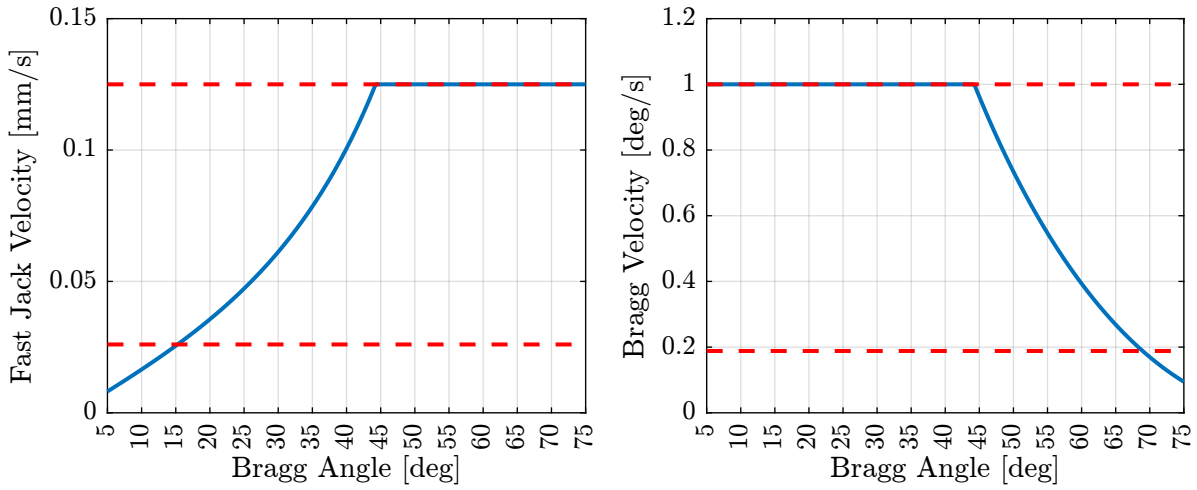

Figure 6.10: Fast jack velocity and Bragg angular velocity during the scan

Important

From Figure 6.10, it is shown that the fast jack velocity as well as the bragg angular velocity are within the bounds except:

- Below 15 degrees where the fast jack velocity is too small. The frequency of the non-linear periodic errors of the interferometers would be at too low frequency (in the pass-band of the filter, see Figure 6.11). One easy option is to use an interferometer without periodic non-linearity. Another option is to increase the maximum Bragg angular velocity to 3 deg/s.
- Above 70 degrees where the Bragg angular velocity is too small. This may introduce low frequency disturbances induced by the `mcoil` motor that would be in the pass-band of the filter (see Figure 6.11). It should be verified if this is indeed problematic of not. An other way would be to scan without the `mcoil` motor at very high bragg angle.

In order to better visualize the filtering problem, the frequency of all the signals are shown as a function of the Bragg angle during the scan in Figure 6.11.

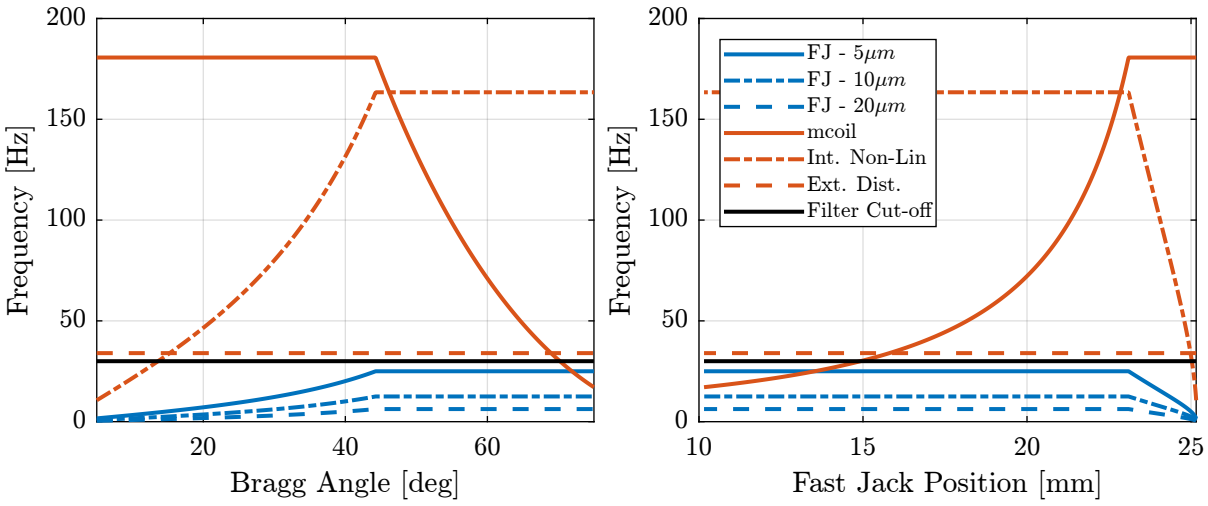

Figure 6.11: Frequency of signals as a function of the Bragg angle and Fast Jack position

7 Piezoelectric LUT

On main issue with implementing the LUT inside

The idea is to:

- Correct the low frequency errors inside the IcePAP (1mm period motor error).
- This should reduce the errors to an acceptable range (within the piezoelectric stack stroke).
- Correct the high frequency errors (periods of $5\,\mu m$, $10\,\mu m$ and $20\,\mu m$) using the piezoelectric stack.